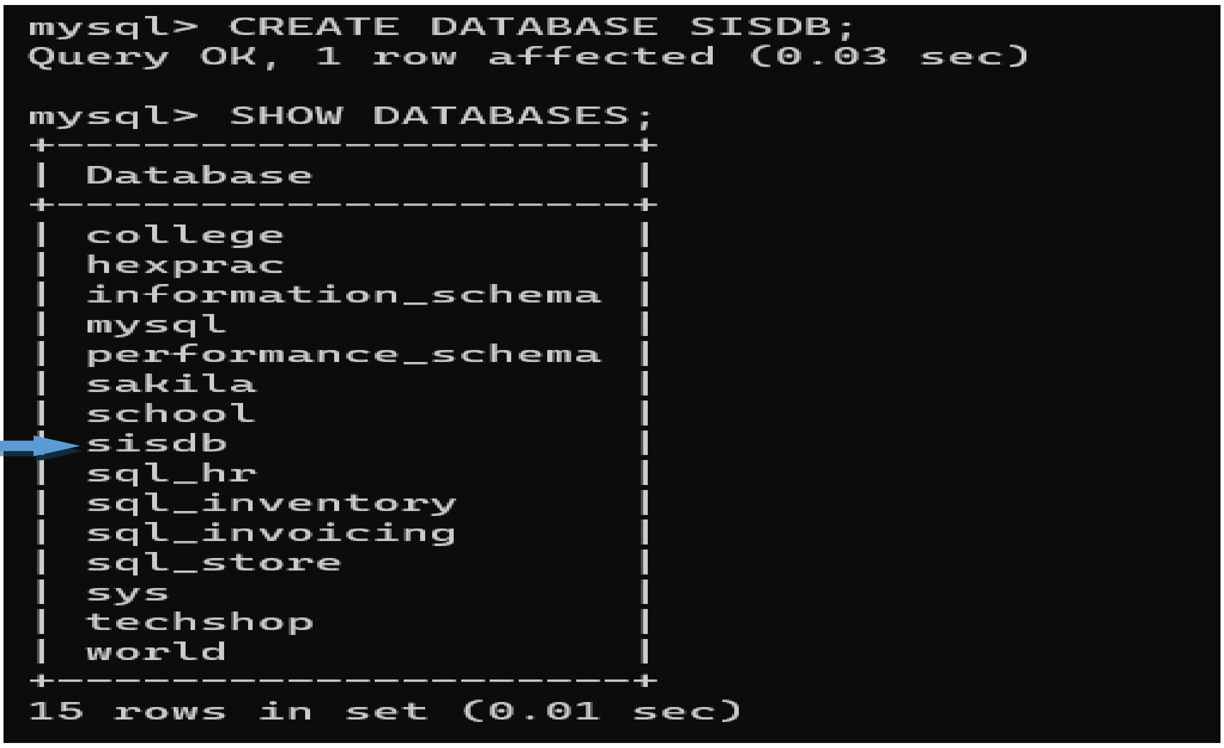
Task 1: Database Design

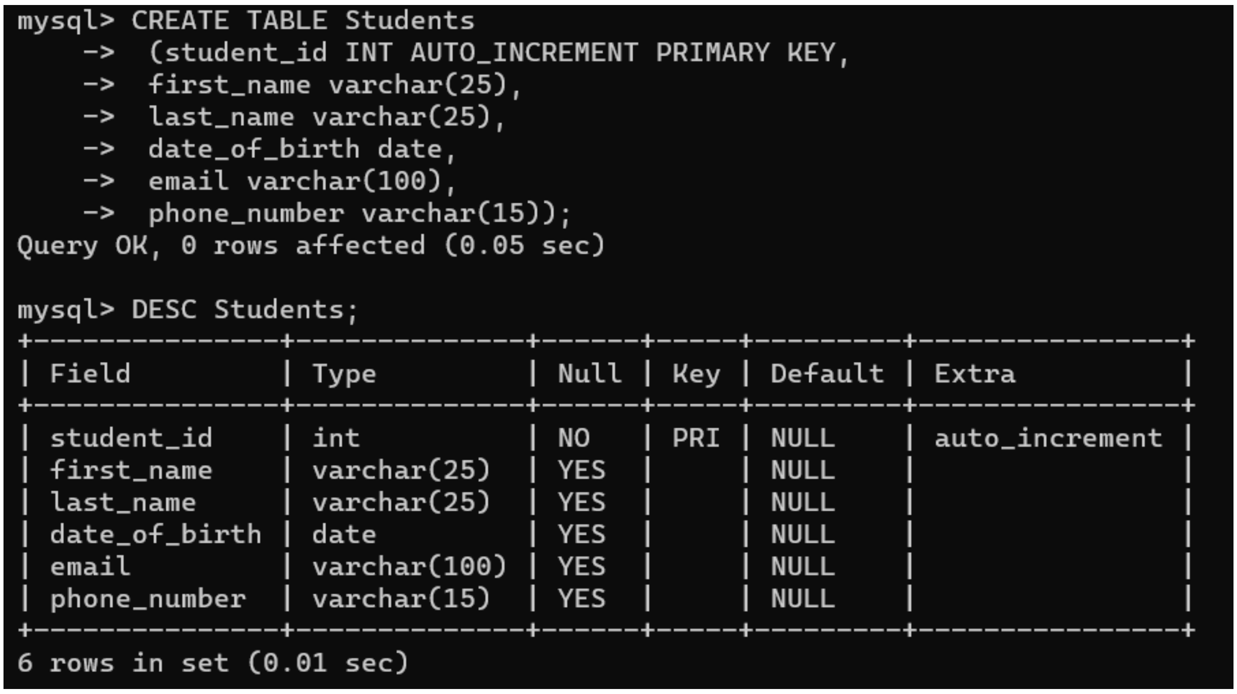
1. Create the database named "SISDB"



```
mysql> CREATE DATABASE SISDB;
Query OK, 1 row affected (0.03 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| college            |
| hexprac            |
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| school             |
| sisdb              |
| sql_hr             |
| sql_inventory      |
| sql_invoicing      |
| sql_store          |
| sys                |
| techshop           |
| world              |
+--------------------+
15 rows in set (0.01 sec)
```

2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

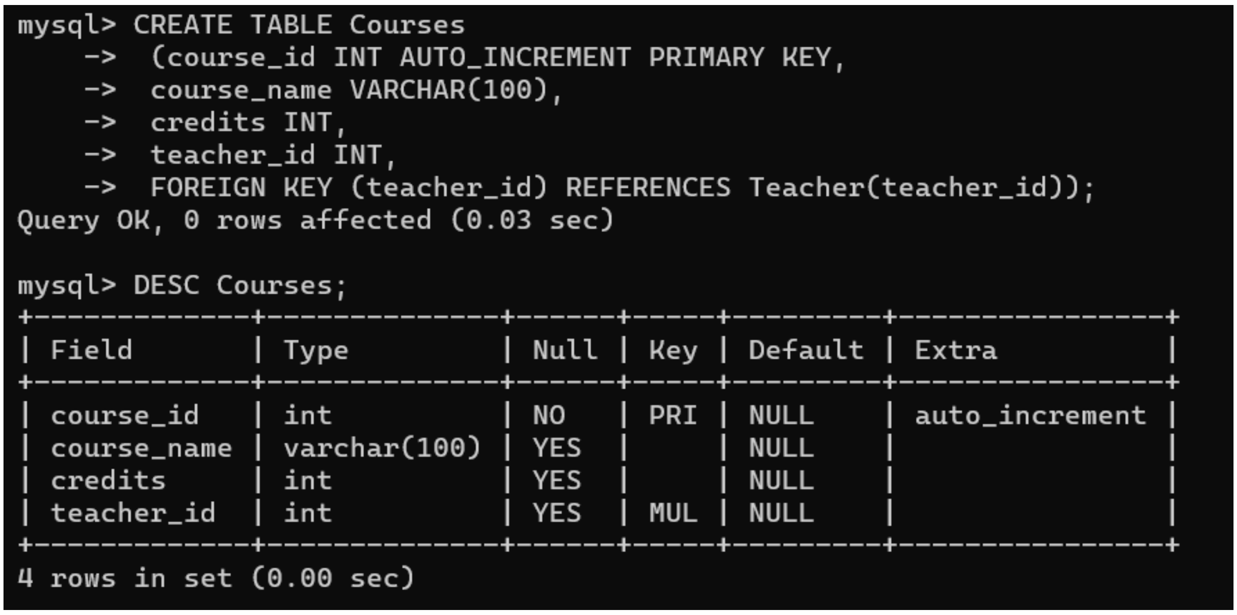
a. Students -



```
mysql> CREATE TABLE Students
    -> (student_id INT AUTO_INCREMENT PRIMARY KEY,
    -> first_name varchar(25),
    -> last_name varchar(25),
    -> date_of_birth date,
    -> email varchar(100),
    -> phone_number varchar(15));
Query OK, 0 rows affected (0.05 sec)

mysql> DESC Students;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| student_id    | int          | NO   | PRI | NULL    | auto_increment |
| first_name    | varchar(25)  | YES  |     | NULL    |                |
| last_name     | varchar(25)  | YES  |     | NULL    |                |
| date_of_birth | date         | YES  |     | NULL    |                |
| email         | varchar(100) | YES  |     | NULL    |                |
| phone_number  | varchar(15)  | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```

(Students table)

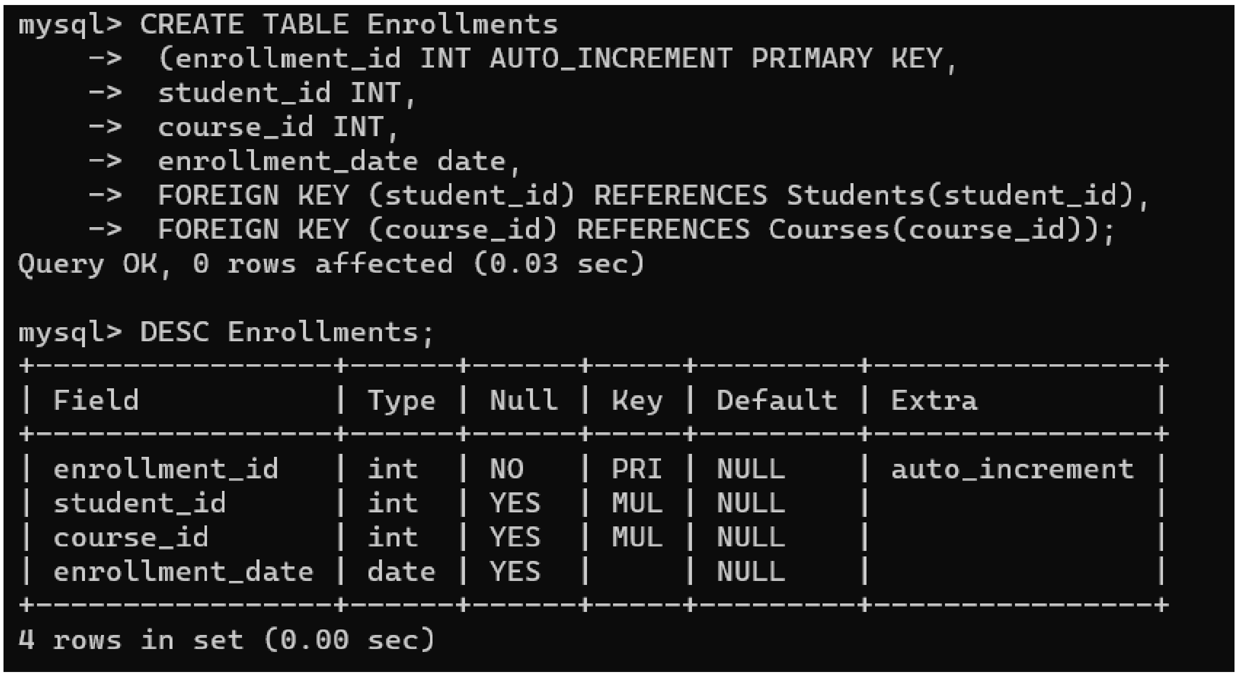
b. Courses



```
mysql> CREATE TABLE Courses
    ->  (course_id INT AUTO_INCREMENT PRIMARY KEY,
    ->  course_name VARCHAR(100),
    ->  credits INT,
    ->  teacher_id INT,
    ->  FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id));
Query OK, 0 rows affected (0.03 sec)

mysql> DESC Courses;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| course_id   | int          | NO   | PRI | NULL    | auto_increment |
| course_name | varchar(100) | YES  |     | NULL    |                |
| credits     | int          | YES  |     | NULL    |                |
| teacher_id  | int          | YES  | MUL | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

(Courses table)

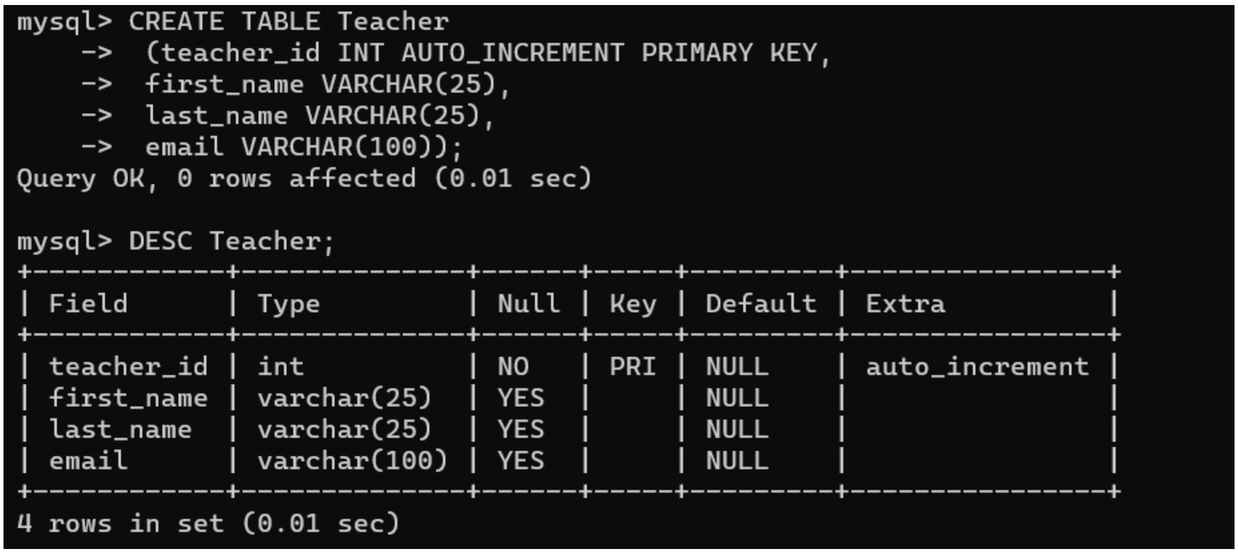
c. Enrollments



```
mysql> CREATE TABLE Enrollments
    ->  (enrollment_id INT AUTO_INCREMENT PRIMARY KEY,
    ->  student_id INT,
    ->  course_id INT,
    ->  enrollment_date date,
    ->  FOREIGN KEY (student_id) REFERENCES Students(student_id),
    ->  FOREIGN KEY (course_id) REFERENCES Courses(course_id));
Query OK, 0 rows affected (0.03 sec)

mysql> DESC Enrollments;
+-----------------+------+------+-----+---------+----------------+
| Field           | Type | Null | Key | Default | Extra          |
+-----------------+------+------+-----+---------+----------------+
| enrollment_id   | int  | NO   | PRI | NULL    | auto_increment |
| student_id      | int  | YES  | MUL | NULL    |                |
| course_id       | int  | YES  | MUL | NULL    |                |
| enrollment_date | date | YES  |     | NULL    |                |
+-----------------+------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

(Enrollments table)

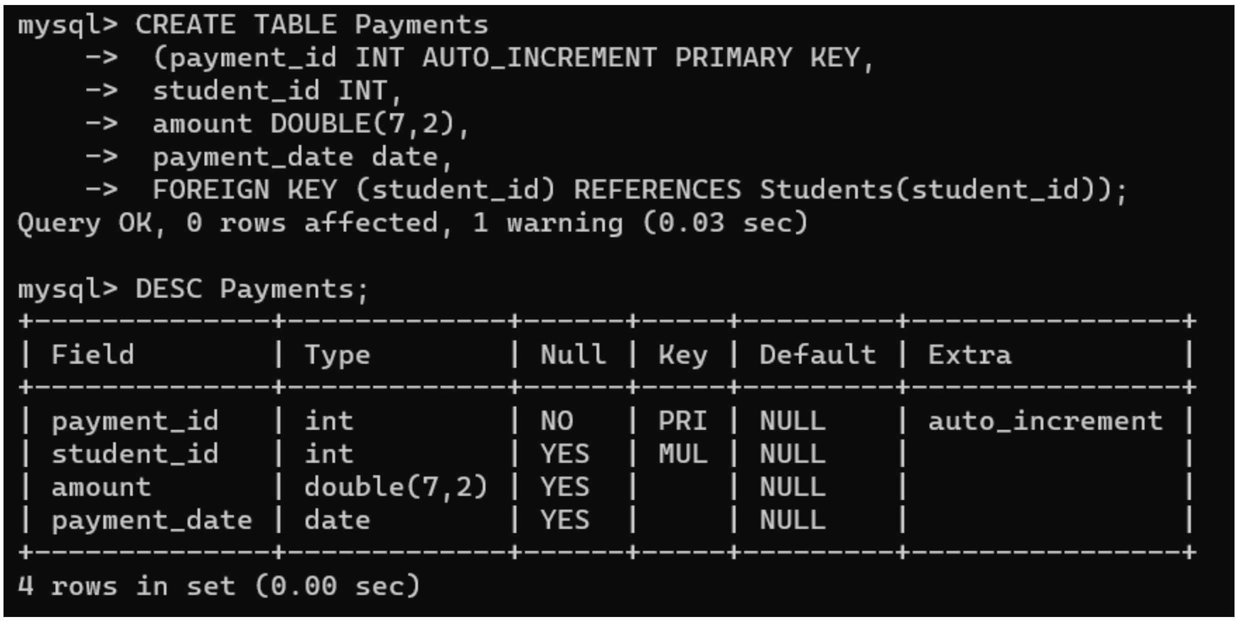
d. Teacher



```
mysql> CREATE TABLE Teacher
    ->  (teacher_id INT AUTO_INCREMENT PRIMARY KEY,
    ->  first_name VARCHAR(25),
    ->  last_name VARCHAR(25),
    ->  email VARCHAR(100));
Query OK, 0 rows affected (0.01 sec)

mysql> DESC Teacher;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| teacher_id | int          | NO   | PRI | NULL    | auto_increment |
| first_name | varchar(25)  | YES  |     | NULL    |                |
| last_name  | varchar(25)  | YES  |     | NULL    |                |
| email      | varchar(100) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

(Teacher table)

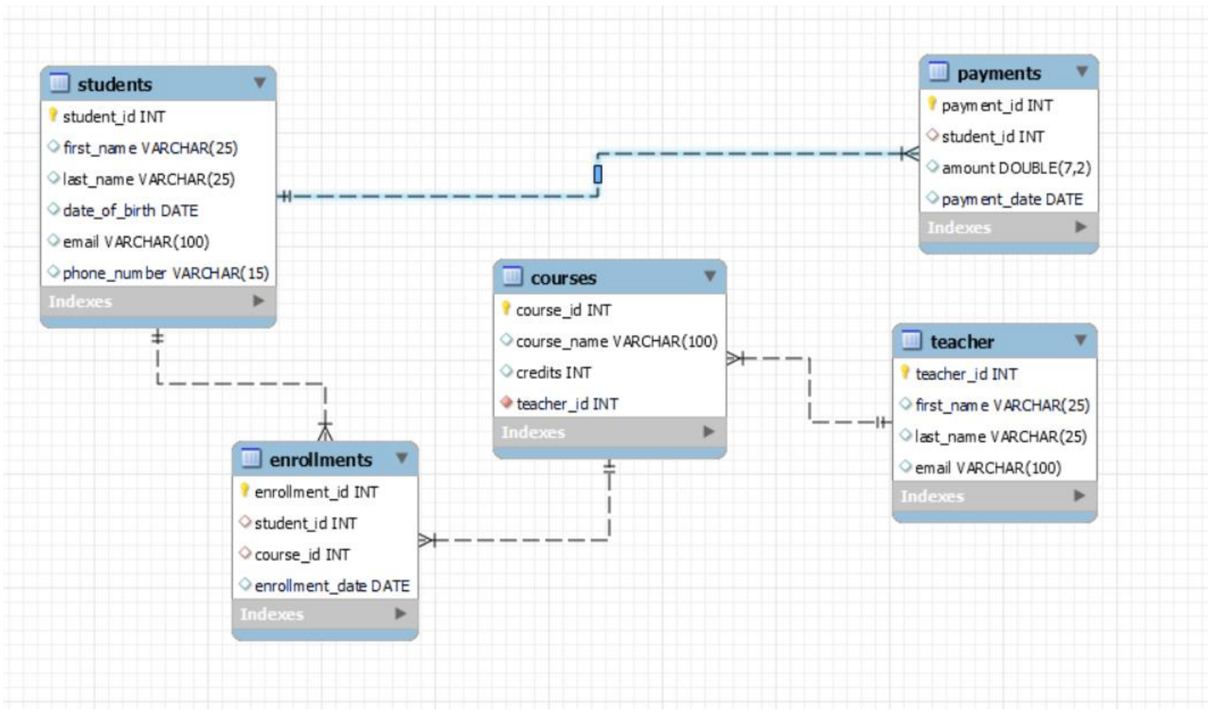
e. Payments



```
mysql> CREATE TABLE Payments
    ->  (payment_id INT AUTO_INCREMENT PRIMARY KEY,
    ->  student_id INT,
    ->  amount DOUBLE(7,2),
    ->  payment_date date,
    ->  FOREIGN KEY (student_id) REFERENCES Students(student_id));
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> DESC Payments;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| payment_id   | int         | NO   | PRI | NULL    | auto_increment |
| student_id   | int         | YES  | MUL | NULL    |                |
| amount       | double(7,2) | YES  |     | NULL    |                |
| payment_date | date        | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

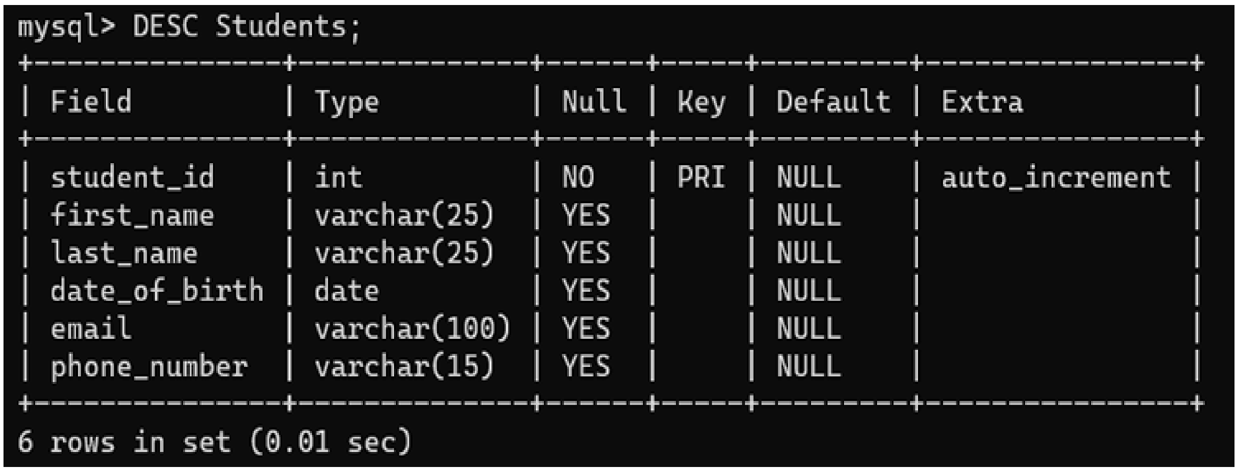
(Payments table)

3. Create an ERD (Entity Relationship Diagram) for the database.



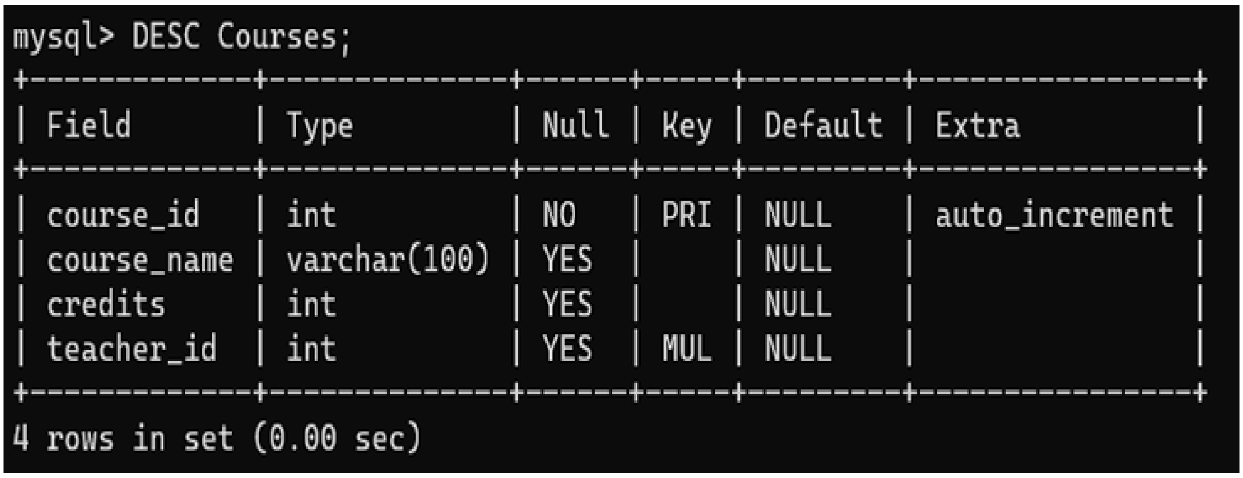
(ERD for SISDB)

4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.



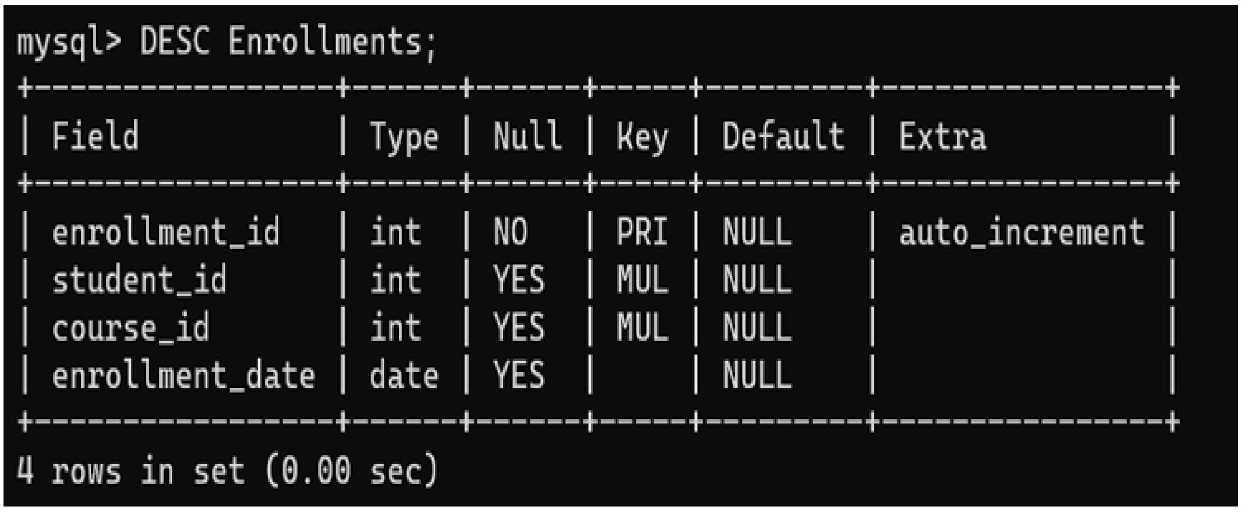
```
mysql> DESC Students;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| student_id    | int          | NO   | PRI | NULL    | auto_increment |
| first_name    | varchar(25)  | YES  |     | NULL    |                |
| last_name     | varchar(25)  | YES  |     | NULL    |                |
| date_of_birth | date         | YES  |     | NULL    |                |
| email         | varchar(100) | YES  |     | NULL    |                |
| phone_number  | varchar(15)  | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```

(Student table)



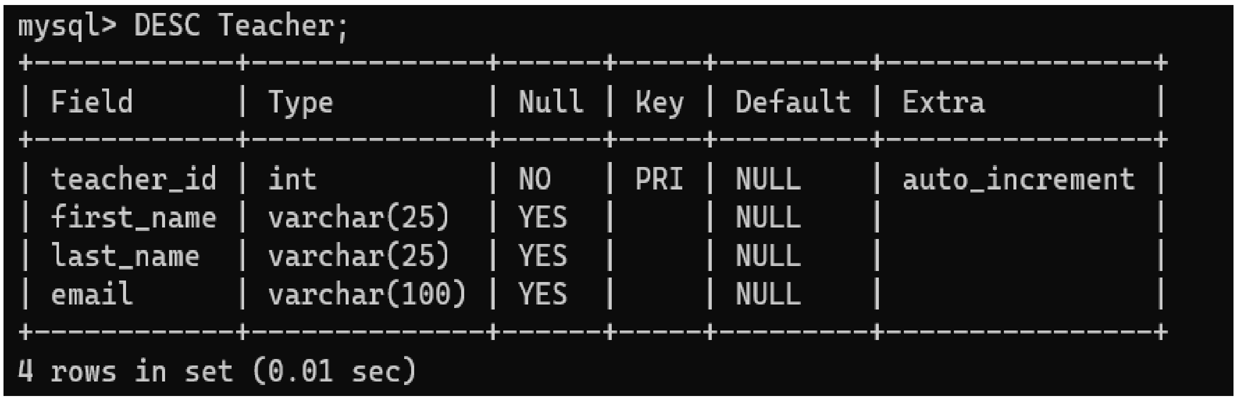
```
mysql> DESC Courses;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| course_id   | int          | NO   | PRI | NULL    | auto_increment |
| course_name | varchar(100) | YES  |     | NULL    |                |
| credits     | int          | YES  |     | NULL    |                |
| teacher_id  | int          | YES  | MUL | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

(Courses table)



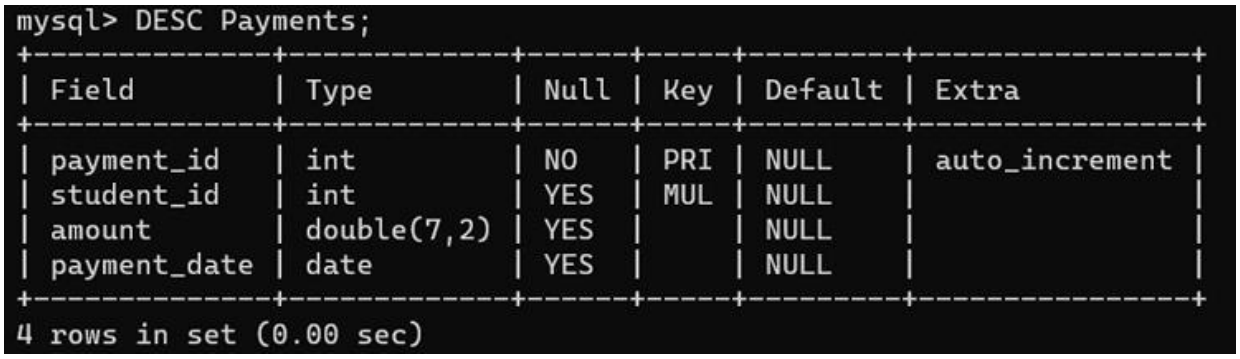
```
mysql> DESC Enrollments;
+-----------------+------+------+-----+---------+----------------+
| Field           | Type | Null | Key | Default | Extra          |
+-----------------+------+------+-----+---------+----------------+
| enrollment_id   | int  | NO   | PRI | NULL    | auto_increment |
| student_id      | int  | YES  | MUL | NULL    |                |
| course_id       | int  | YES  | MUL | NULL    |                |
| enrollment_date | date | YES  |     | NULL    |                |
+-----------------+------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

(Enrollments table)



```
mysql> DESC Teacher;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| teacher_id | int          | NO   | PRI | NULL    | auto_increment |
| first_name | varchar(25)  | YES  |     | NULL    |                |
| last_name  | varchar(25)  | YES  |     | NULL    |                |
| email      | varchar(100) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

(Teacher table)

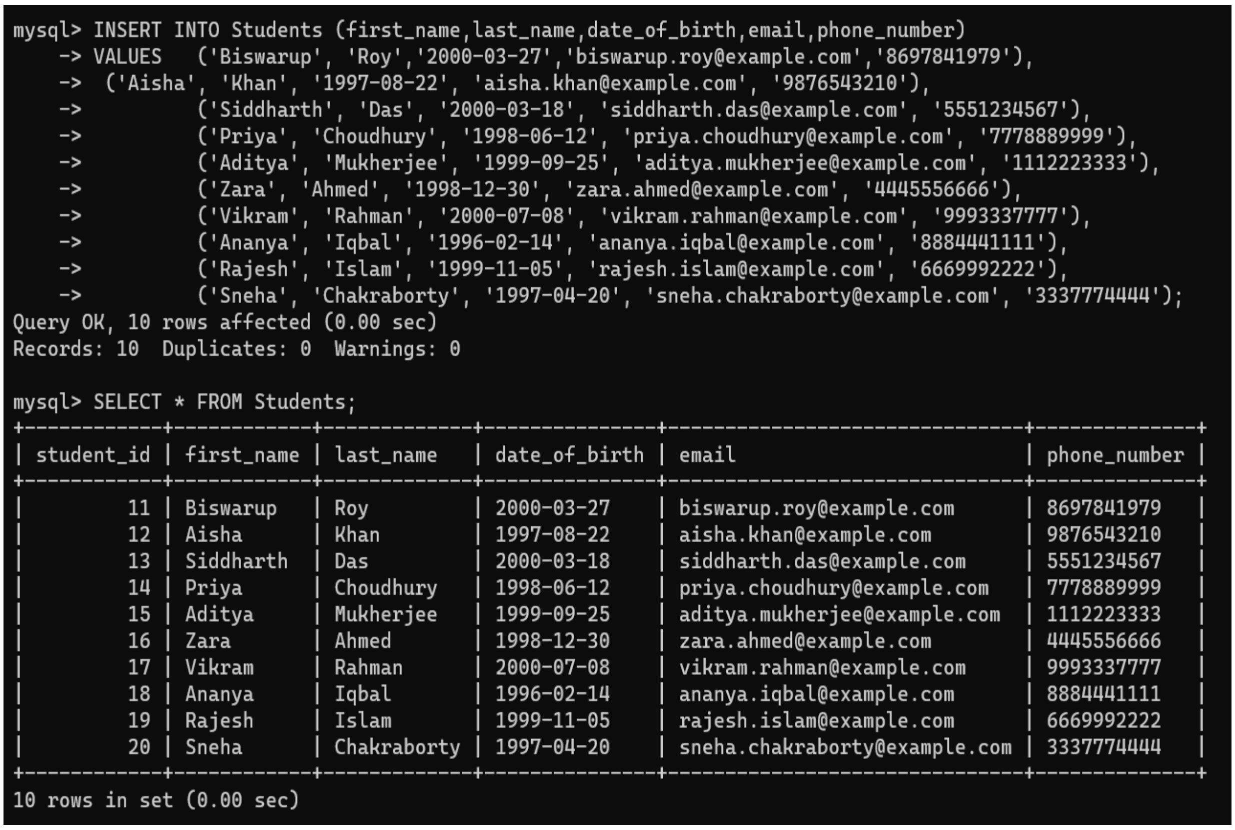


```
mysql> DESC Payments;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| payment_id   | int         | NO   | PRI | NULL    | auto_increment |
| student_id   | int         | YES  | MUL | NULL    |                |
| amount       | double(7,2) | YES  |     | NULL    |                |
| payment_date | date        | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

(Payments table)

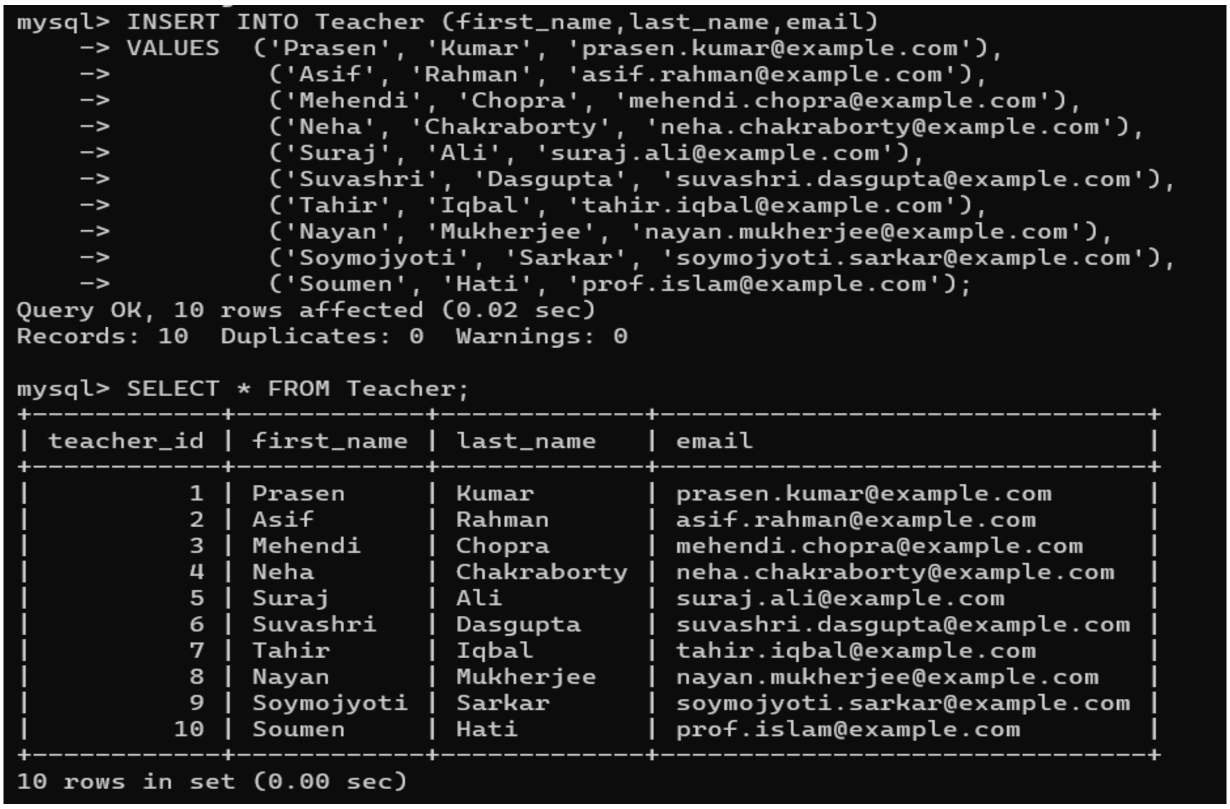
5. Insert at least 10 sample records into each of the following tables.

i. Students



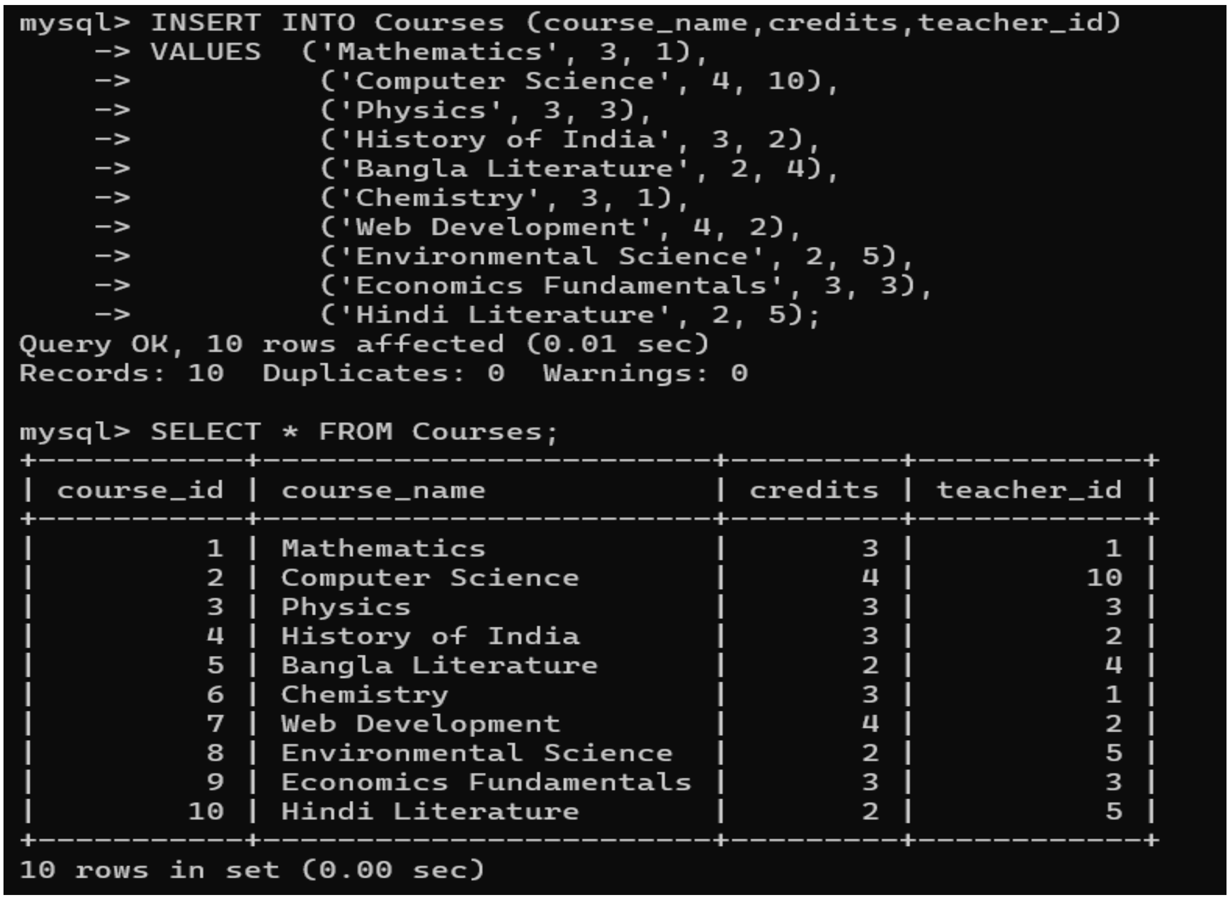
```
mysql> INSERT INTO Students (first_name,last_name,date_of_birth,email,phone_number)
    -> VALUES   ('Biswarup', 'Roy','2000-03-27','biswarup.roy@example.com','8697841979'),
    ->  ('Aisha', 'Khan', '1997-08-22', 'aisha.khan@example.com', '9876543210'),
    ->         ('Siddharth', 'Das', '2000-03-18', 'siddharth.das@example.com', '5551234567'),
    ->         ('Priya', 'Choudhury', '1998-06-12', 'priya.choudhury@example.com', '7778889999'),
    ->         ('Aditya', 'Mukherjee', '1999-09-25', 'aditya.mukherjee@example.com', '1112223333'),
    ->         ('Zara', 'Ahmed', '1998-12-30', 'zara.ahmed@example.com', '4445556666'),
    ->         ('Vikram', 'Rahman', '2000-07-08', 'vikram.rahman@example.com', '9993337777'),
    ->         ('Ananya', 'Iqbal', '1996-02-14', 'ananya.iqbal@example.com', '8884441111'),
    ->         ('Rajesh', 'Islam', '1999-11-05', 'rajesh.islam@example.com', '6669992222'),
    ->         ('Sneha', 'Chakraborty', '1997-04-20', 'sneha.chakraborty@example.com', '3337774444');
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Students;
+------------+------------+-------------+---------------+-------------------------------+--------------+
| student_id | first_name | last_name   | date_of_birth | email                         | phone_number |
+------------+------------+-------------+---------------+-------------------------------+--------------+
|         11 | Biswarup   | Roy         | 2000-03-27    | biswarup.roy@example.com      | 8697841979   |
|         12 | Aisha      | Khan        | 1997-08-22    | aisha.khan@example.com        | 9876543210   |
|         13 | Siddharth  | Das         | 2000-03-18    | siddharth.das@example.com     | 5551234567   |
|         14 | Priya      | Choudhury   | 1998-06-12    | priya.choudhury@example.com   | 7778889999   |
|         15 | Aditya     | Mukherjee   | 1999-09-25    | aditya.mukherjee@example.com  | 1112223333   |
|         16 | Zara       | Ahmed       | 1998-12-30    | zara.ahmed@example.com        | 4445556666   |
|         17 | Vikram     | Rahman      | 2000-07-08    | vikram.rahman@example.com     | 9993337777   |
|         18 | Ananya     | Iqbal       | 1996-02-14    | ananya.iqbal@example.com      | 8884441111   |
|         19 | Rajesh     | Islam       | 1999-11-05    | rajesh.islam@example.com      | 6669992222   |
|         20 | Sneha      | Chakraborty | 1997-04-20    | sneha.chakraborty@example.com | 3337774444   |
+------------+------------+-------------+---------------+-------------------------------+--------------+
10 rows in set (0.00 sec)
```

ii. Teacher



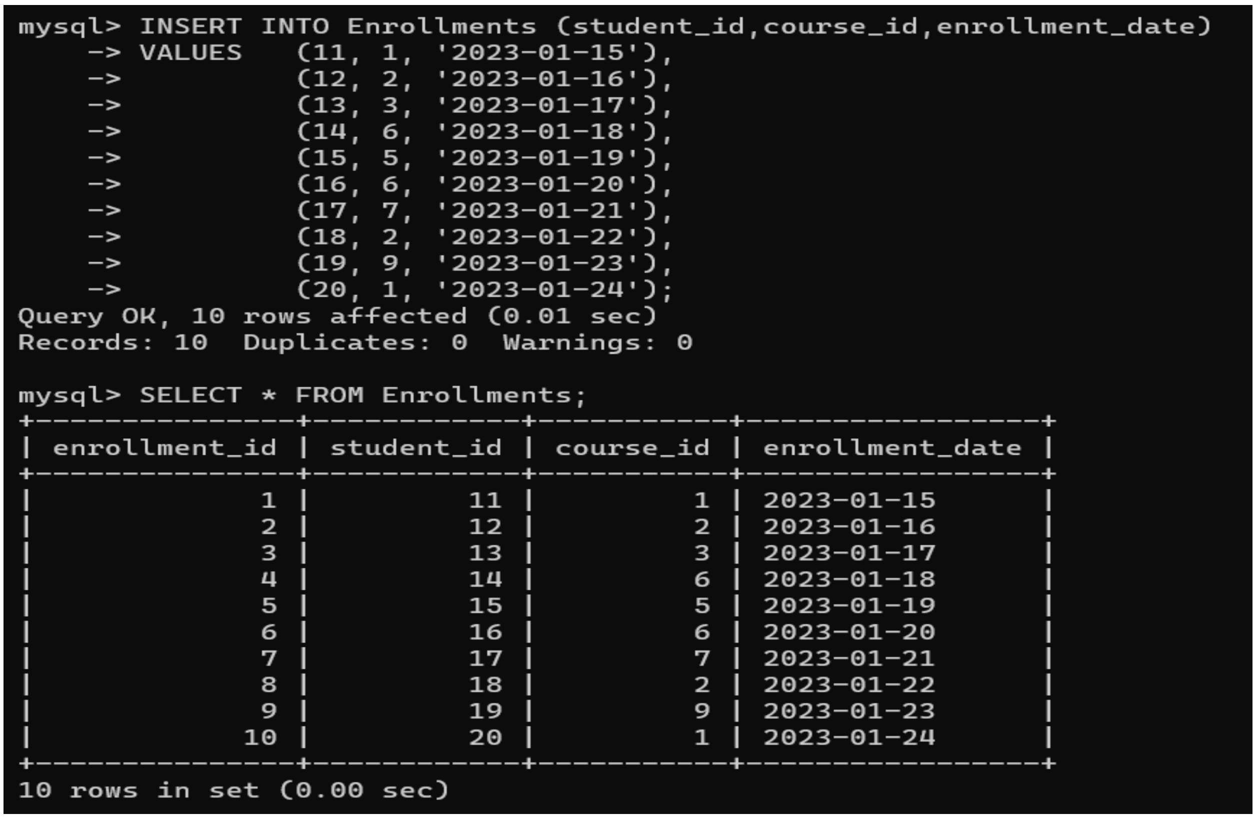
```
mysql> INSERT INTO Teacher (first_name,last_name,email)
    -> VALUES  ('Prasen', 'Kumar', 'prasen.kumar@example.com'),
    ->          ('Asif', 'Rahman', 'asif.rahman@example.com'),
    ->          ('Mehendi', 'Chopra', 'mehendi.chopra@example.com'),
    ->          ('Neha', 'Chakraborty', 'neha.chakraborty@example.com'),
    ->          ('Suraj', 'Ali', 'suraj.ali@example.com'),
    ->          ('Suvashri', 'Dasgupta', 'suvashri.dasgupta@example.com'),
    ->          ('Tahir', 'Iqbal', 'tahir.iqbal@example.com'),
    ->          ('Nayan', 'Mukherjee', 'nayan.mukherjee@example.com'),
    ->          ('Soymojyoti', 'Sarkar', 'soymojyoti.sarkar@example.com'),
    ->          ('Soumen', 'Hati', 'prof.islam@example.com');
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Teacher;
+------------+------------+-------------+-------------------------------+
| teacher_id | first_name | last_name   | email                         |
+------------+------------+-------------+-------------------------------+
|          1 | Prasen     | Kumar       | prasen.kumar@example.com      |
|          2 | Asif       | Rahman      | asif.rahman@example.com       |
|          3 | Mehendi    | Chopra      | mehendi.chopra@example.com    |
|          4 | Neha       | Chakraborty | neha.chakraborty@example.com  |
|          5 | Suraj      | Ali         | suraj.ali@example.com         |
|          6 | Suvashri   | Dasgupta    | suvashri.dasgupta@example.com |
|          7 | Tahir      | Iqbal       | tahir.iqbal@example.com       |
|          8 | Nayan      | Mukherjee   | nayan.mukherjee@example.com   |
|          9 | Soymojyoti | Sarkar      | soymojyoti.sarkar@example.com |
|         10 | Soumen     | Hati        | prof.islam@example.com        |
+------------+------------+-------------+-------------------------------+
10 rows in set (0.00 sec)
```

iii. Courses



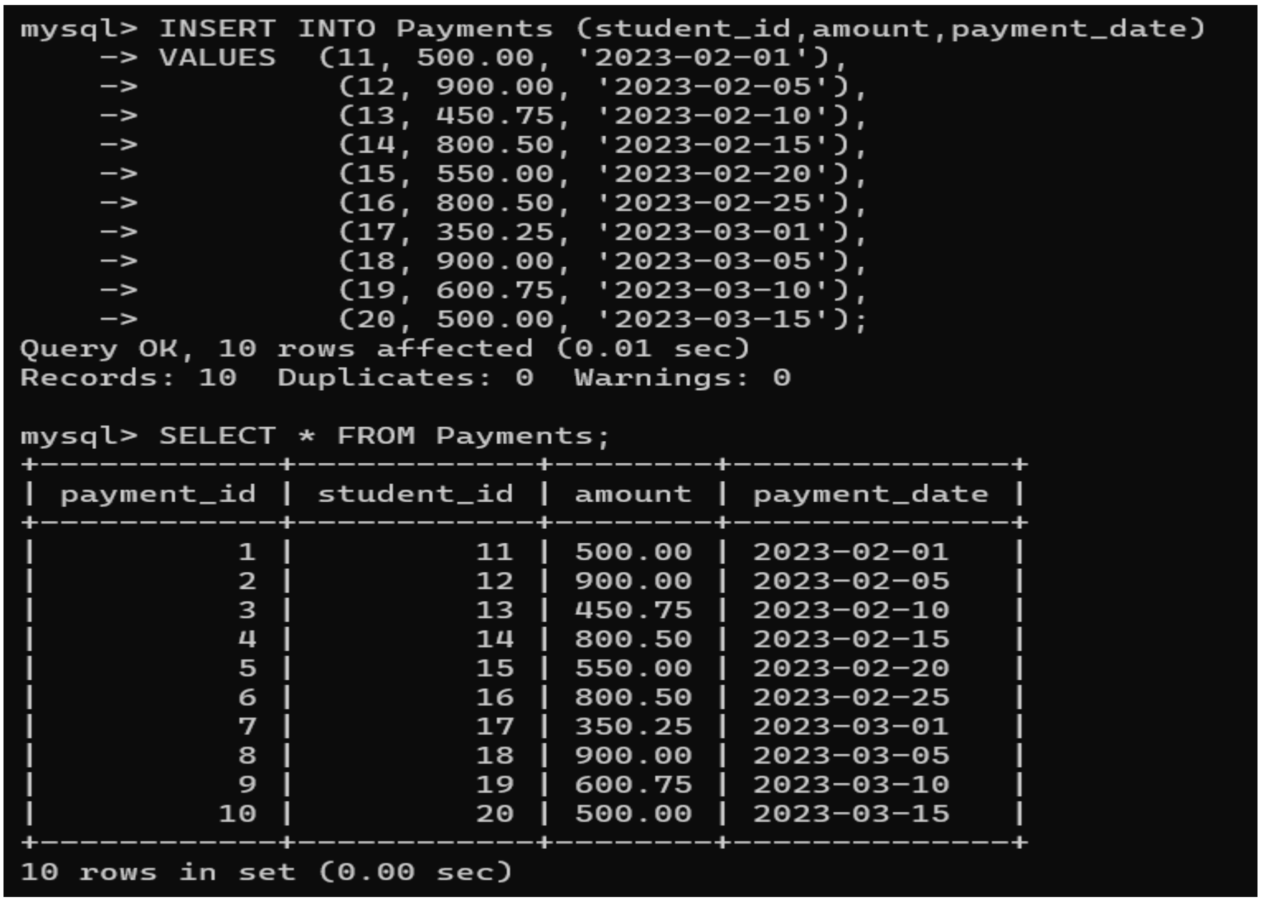
```
mysql> INSERT INTO Courses (course_name,credits,teacher_id)
    -> VALUES  ('Mathematics', 3, 1),
    ->          ('Computer Science', 4, 10),
    ->          ('Physics', 3, 3),
    ->          ('History of India', 3, 2),
    ->          ('Bangla Literature', 2, 4),
    ->          ('Chemistry', 3, 1),
    ->          ('Web Development', 4, 2),
    ->          ('Environmental Science', 2, 5),
    ->          ('Economics Fundamentals', 3, 3),
    ->          ('Hindi Literature', 2, 5);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Courses;
+-----------+------------------------+---------+------------+
| course_id | course_name            | credits | teacher_id |
+-----------+------------------------+---------+------------+
|         1 | Mathematics            |       3 |          1 |
|         2 | Computer Science       |       4 |         10 |
|         3 | Physics                |       3 |          3 |
|         4 | History of India       |       3 |          2 |
|         5 | Bangla Literature      |       2 |          4 |
|         6 | Chemistry              |       3 |          1 |
|         7 | Web Development        |       4 |          2 |
|         8 | Environmental Science  |       2 |          5 |
|         9 | Economics Fundamentals |       3 |          3 |
|        10 | Hindi Literature       |       2 |          5 |
+-----------+------------------------+---------+------------+
10 rows in set (0.00 sec)
```

iv. Enrollments



```
mysql> INSERT INTO Enrollments (student_id,course_id,enrollment_date)
    -> VALUES   (11, 1, '2023-01-15'),
    ->          (12, 2, '2023-01-16'),
    ->          (13, 3, '2023-01-17'),
    ->          (14, 6, '2023-01-18'),
    ->          (15, 5, '2023-01-19'),
    ->          (16, 6, '2023-01-20'),
    ->          (17, 7, '2023-01-21'),
    ->          (18, 2, '2023-01-22'),
    ->          (19, 9, '2023-01-23'),
    ->          (20, 1, '2023-01-24');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|             1 |         11 |         1 | 2023-01-15      |
|             2 |         12 |         2 | 2023-01-16      |
|             3 |         13 |         3 | 2023-01-17      |
|             4 |         14 |         6 | 2023-01-18      |
|             5 |         15 |         5 | 2023-01-19      |
|             6 |         16 |         6 | 2023-01-20      |
|             7 |         17 |         7 | 2023-01-21      |
|             8 |         18 |         2 | 2023-01-22      |
|             9 |         19 |         9 | 2023-01-23      |
|            10 |         20 |         1 | 2023-01-24      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```

v. Payments

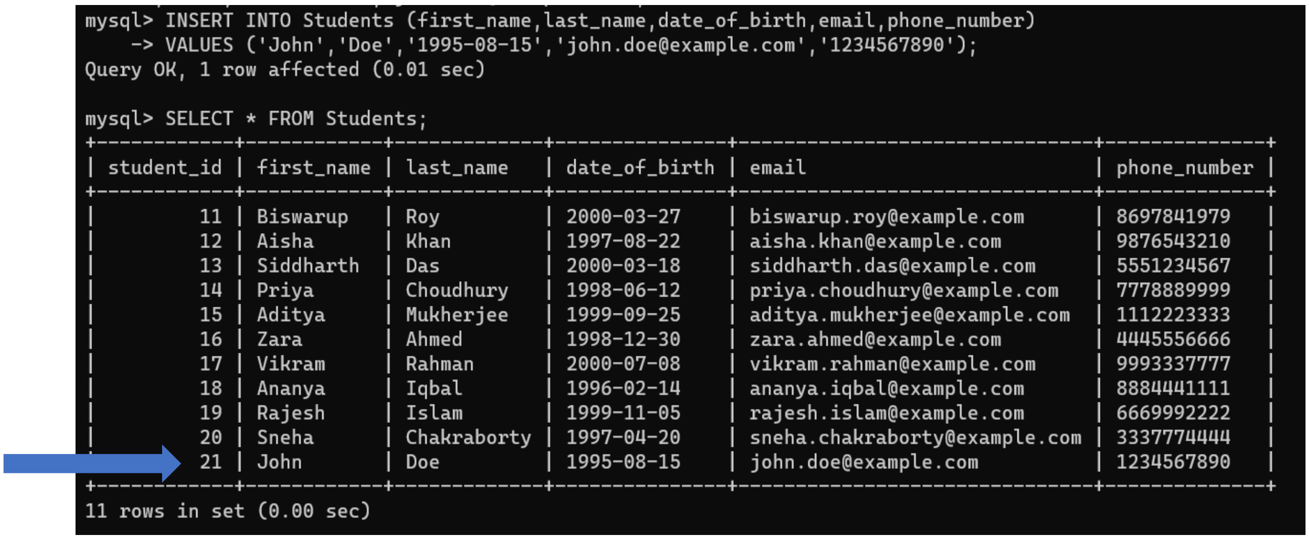


```
mysql> INSERT INTO Payments (student_id,amount,payment_date)
    -> VALUES  (11, 500.00, '2023-02-01'),
    ->          (12, 900.00, '2023-02-05'),
    ->          (13, 450.75, '2023-02-10'),
    ->          (14, 800.50, '2023-02-15'),
    ->          (15, 550.00, '2023-02-20'),
    ->          (16, 800.50, '2023-02-25'),
    ->          (17, 350.25, '2023-03-01'),
    ->          (18, 900.00, '2023-03-05'),
    ->          (19, 600.75, '2023-03-10'),
    ->          (20, 500.00, '2023-03-15');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|          1 |         11 | 500.00 | 2023-02-01   |
|          2 |         12 | 900.00 | 2023-02-05   |
|          3 |         13 | 450.75 | 2023-02-10   |
|          4 |         14 | 800.50 | 2023-02-15   |
|          5 |         15 | 550.00 | 2023-02-20   |
|          6 |         16 | 800.50 | 2023-02-25   |
|          7 |         17 | 350.25 | 2023-03-01   |
|          8 |         18 | 900.00 | 2023-03-05   |
|          9 |         19 | 600.75 | 2023-03-10   |
|         10 |         20 | 500.00 | 2023-03-15   |
+------------+------------+--------+--------------+
10 rows in set (0.00 sec)
```

TASK 2: Select, Where, Between, AND, LIKE

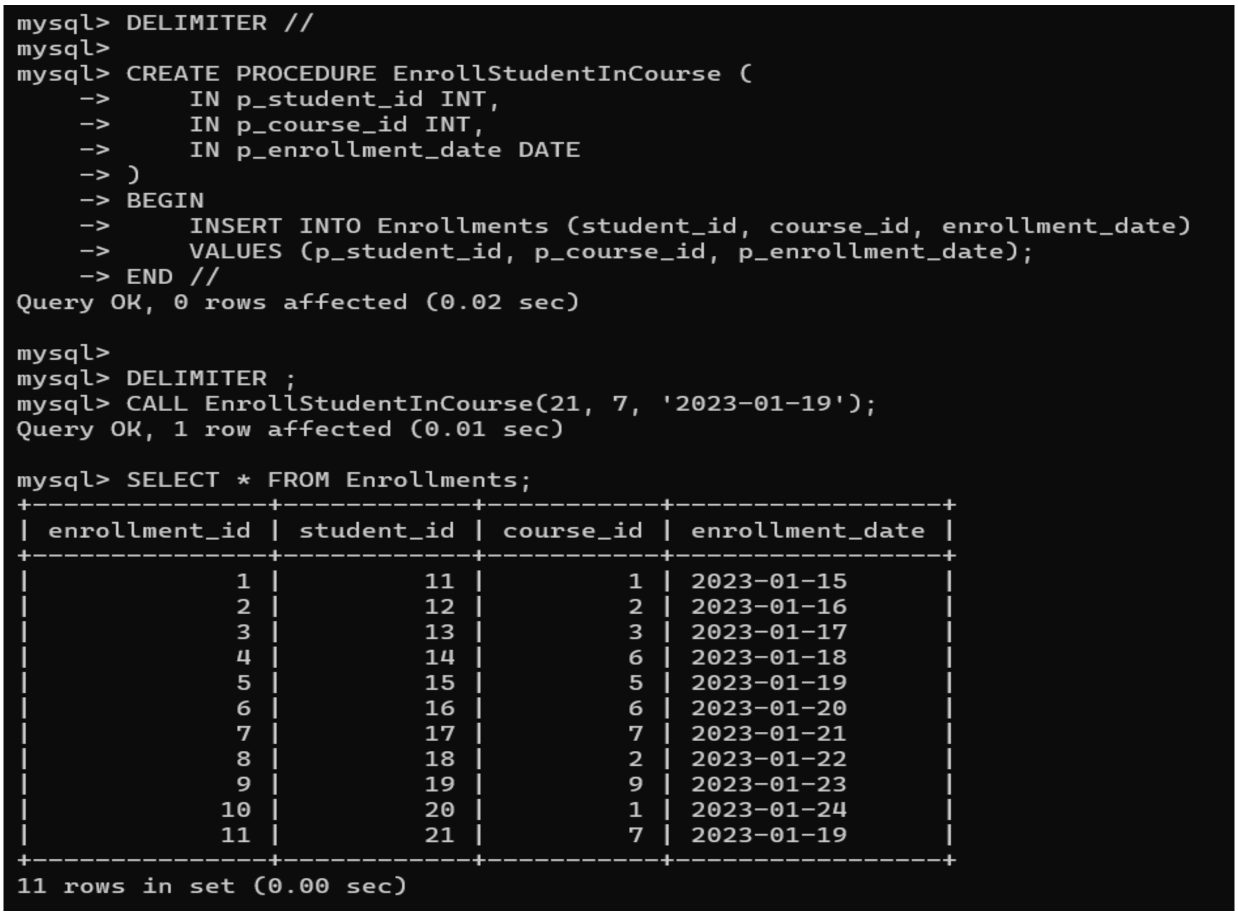
1. Write an SQL query to insert a new student into the "Students" table with the following details:

(a)First Name: John (b)Last Name: Doe (c)Date of Birth: 1995-08-15
(d)Email:john.doe@example.com (e) Phone Number: 1234567890



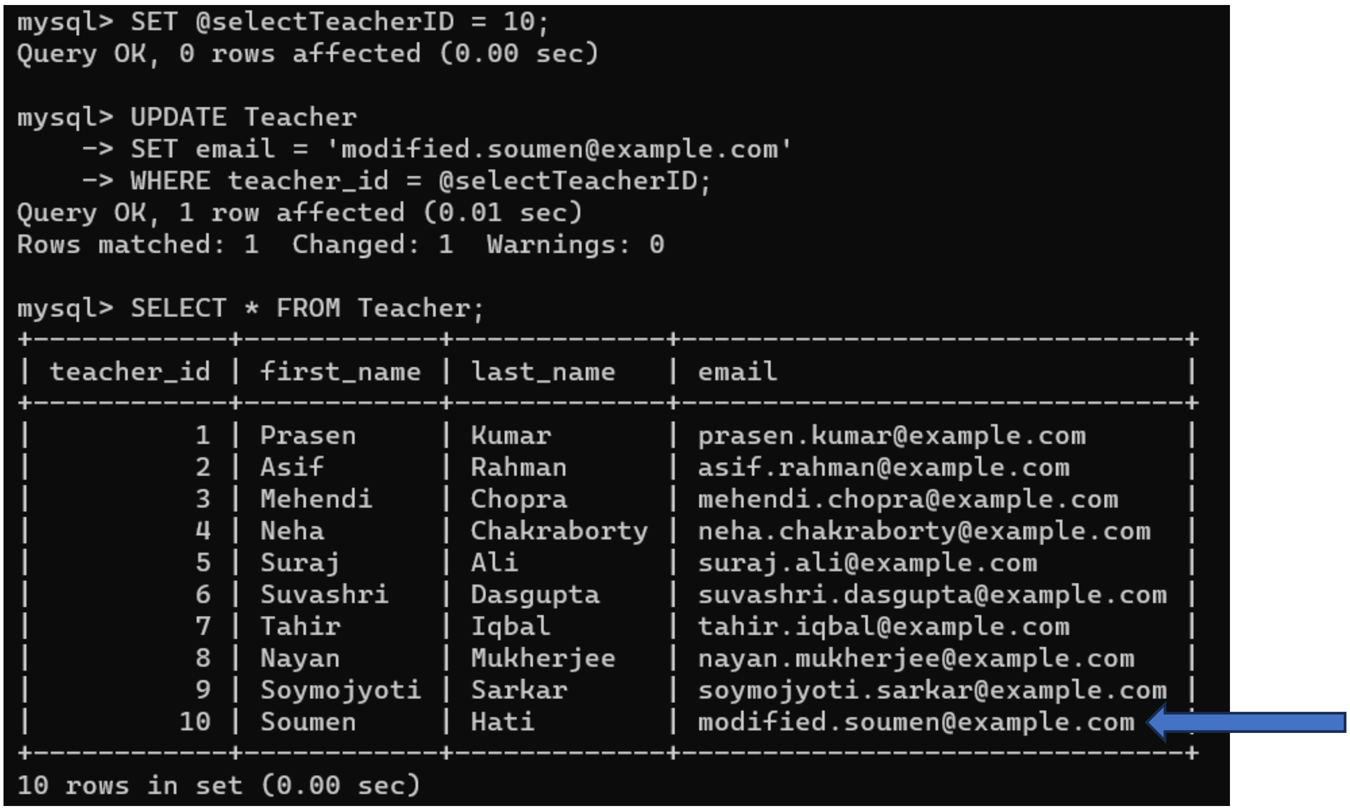
```
mysql> INSERT INTO Students (first_name,last_name,date_of_birth,email,phone_number)
    -> VALUES ('John','Doe','1995-08-15','john.doe@example.com','1234567890');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Students;
+------------+------------+-------------+---------------+------------------------------+--------------+
| student_id | first_name | last_name   | date_of_birth | email                        | phone_number |
+------------+------------+-------------+---------------+------------------------------+--------------+
|         11 | Biswarup   | Roy         | 2000-03-27    | biswarup.roy@example.com     | 8697841979   |
|         12 | Aisha      | Khan        | 1997-08-22    | aisha.khan@example.com       | 9876543210   |
|         13 | Siddharth  | Das         | 2000-03-18    | siddharth.das@example.com    | 5551234567   |
|         14 | Priya      | Choudhury   | 1998-06-12    | priya.choudhury@example.com  | 7778889999   |
|         15 | Aditya     | Mukherjee   | 1999-09-25    | aditya.mukherjee@example.com | 1112223333   |
|         16 | Zara       | Ahmed       | 1998-12-30    | zara.ahmed@example.com       | 4445556666   |
|         17 | Vikram     | Rahman      | 2000-07-08    | vikram.rahman@example.com    | 9993337777   |
|         18 | Ananya     | Iqbal       | 1996-02-14    | ananya.iqbal@example.com     | 8884441111   |
|         19 | Rajesh     | Islam       | 1999-11-05    | rajesh.islam@example.com     | 6669992222   |
|         20 | Sneha      | Chakraborty | 1997-04-20    | sneha.chakraborty@example.com | 3337774444  |
|         21 | John       | Doe         | 1995-08-15    | john.doe@example.com         | 1234567890   |
+------------+------------+-------------+---------------+------------------------------+--------------+
11 rows in set (0.00 sec)
```

2. Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.



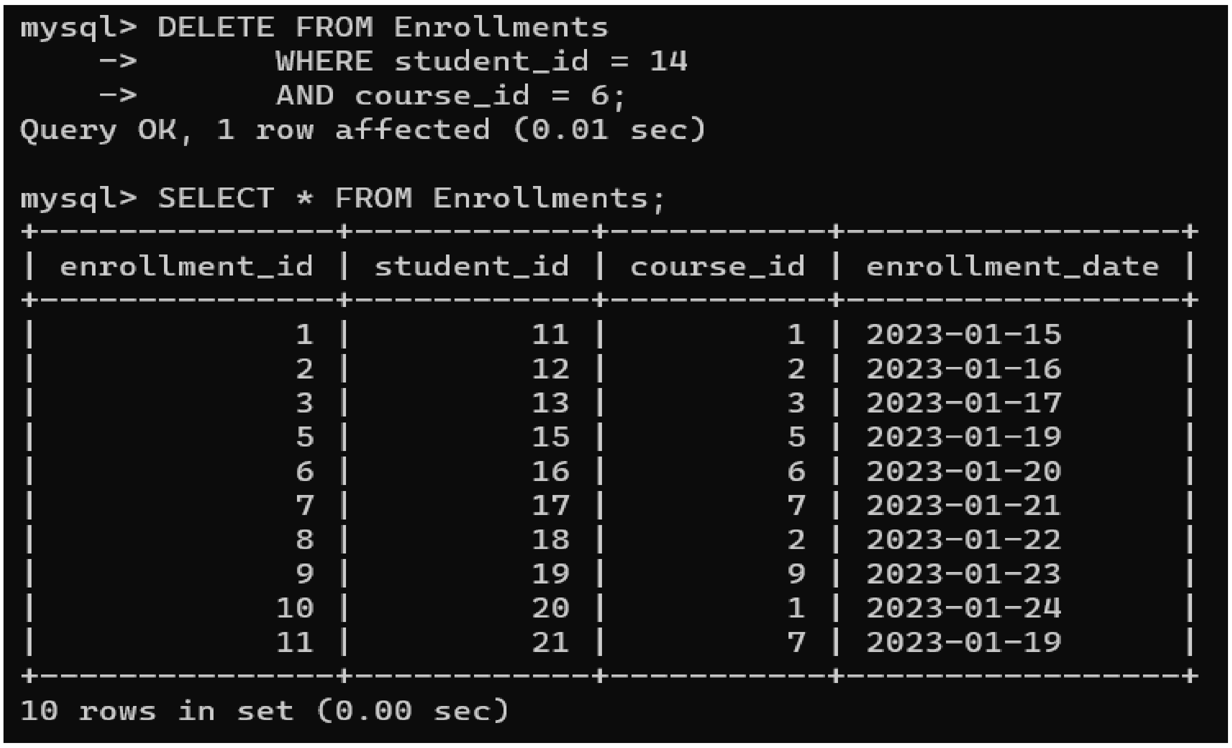
```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE EnrollStudentInCourse (
    ->     IN p_student_id INT,
    ->     IN p_course_id INT,
    ->     IN p_enrollment_date DATE
    -> )
    -> BEGIN
    ->     INSERT INTO Enrollments (student_id, course_id, enrollment_date)
    ->     VALUES (p_student_id, p_course_id, p_enrollment_date);
    -> END //
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL EnrollStudentInCourse(21, 7, '2023-01-19');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|             1 |         11 |         1 | 2023-01-15      |
|             2 |         12 |         2 | 2023-01-16      |
|             3 |         13 |         3 | 2023-01-17      |
|             4 |         14 |         6 | 2023-01-18      |
|             5 |         15 |         5 | 2023-01-19      |
|             6 |         16 |         6 | 2023-01-20      |
|             7 |         17 |         7 | 2023-01-21      |
|             8 |         18 |         2 | 2023-01-22      |
|             9 |         19 |         9 | 2023-01-23      |
|            10 |         20 |         1 | 2023-01-24      |
|            11 |         21 |         7 | 2023-01-19      |
+---------------+------------+-----------+-----------------+
11 rows in set (0.00 sec)
```

3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.



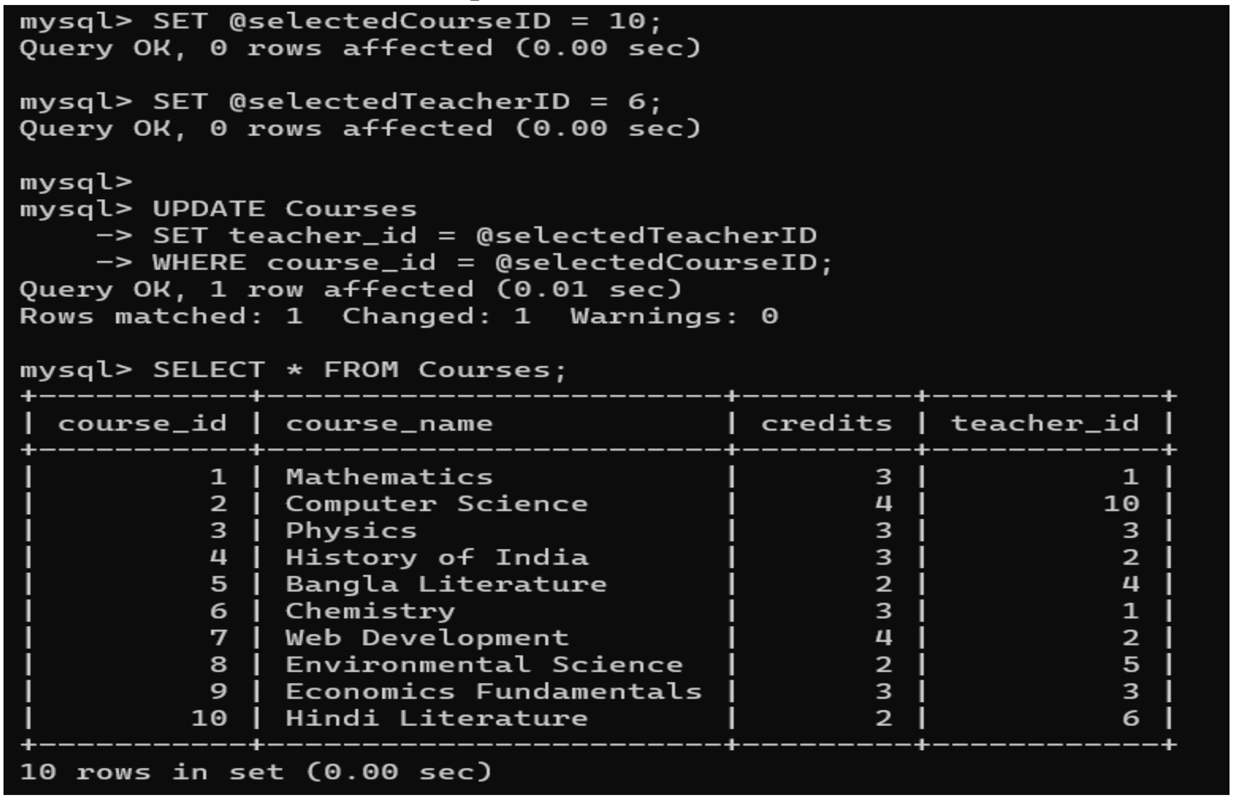
```
mysql> SET @selectTeacherID = 10;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE Teacher
    -> SET email = 'modified.soumen@example.com'
    -> WHERE teacher_id = @selectTeacherID;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Teacher;
+------------+------------+-------------+-------------------------------+
| teacher_id | first_name | last_name   | email                         |
+------------+------------+-------------+-------------------------------+
|          1 | Prasen     | Kumar       | prasen.kumar@example.com      |
|          2 | Asif       | Rahman      | asif.rahman@example.com       |
|          3 | Mehendi    | Chopra      | mehendi.chopra@example.com    |
|          4 | Neha       | Chakraborty | neha.chakraborty@example.com  |
|          5 | Suraj      | Ali         | suraj.ali@example.com         |
|          6 | Suvashri   | Dasgupta    | suvashri.dasgupta@example.com |
|          7 | Tahir      | Iqbal       | tahir.iqbal@example.com       |
|          8 | Nayan      | Mukherjee   | nayan.mukherjee@example.com   |
|          9 | Soymojyoti | Sarkar      | soymojyoti.sarkar@example.com |
|         10 | Soumen     | Hati        | modified.soumen@example.com   |
+------------+------------+-------------+-------------------------------+
10 rows in set (0.00 sec)
```

4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.



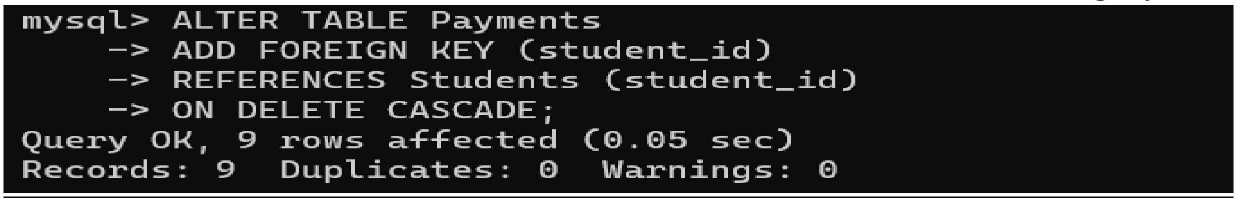
```
mysql> DELETE FROM Enrollments
    ->       WHERE student_id = 14
    ->       AND course_id = 6;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|             1 |         11 |         1 | 2023-01-15      |
|             2 |         12 |         2 | 2023-01-16      |
|             3 |         13 |         3 | 2023-01-17      |
|             5 |         15 |         5 | 2023-01-19      |
|             6 |         16 |         6 | 2023-01-20      |
|             7 |         17 |         7 | 2023-01-21      |
|             8 |         18 |         2 | 2023-01-22      |
|             9 |         19 |         9 | 2023-01-23      |
|            10 |         20 |         1 | 2023-01-24      |
|            11 |         21 |         7 | 2023-01-19      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```

5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.

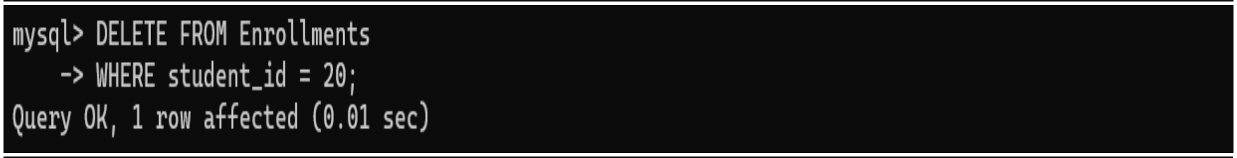


```
mysql> SET @selectedCourseID = 10;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @selectedTeacherID = 6;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> UPDATE Courses
    -> SET teacher_id = @selectedTeacherID
    -> WHERE course_id = @selectedCourseID;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Courses;
+-----------+------------------------+---------+------------+
| course_id | course_name            | credits | teacher_id |
+-----------+------------------------+---------+------------+
|         1 | Mathematics            |       3 |          1 |
|         2 | Computer Science       |       4 |         10 |
|         3 | Physics                |       3 |          3 |
|         4 | History of India       |       3 |          2 |
|         5 | Bangla Literature      |       2 |          4 |
|         6 | Chemistry              |       3 |          1 |
|         7 | Web Development        |       4 |          2 |
|         8 | Environmental Science  |       2 |          5 |
|         9 | Economics Fundamentals |       3 |          3 |
|        10 | Hindi Literature       |       2 |          6 |
+-----------+------------------------+---------+------------+
10 rows in set (0.00 sec)
```

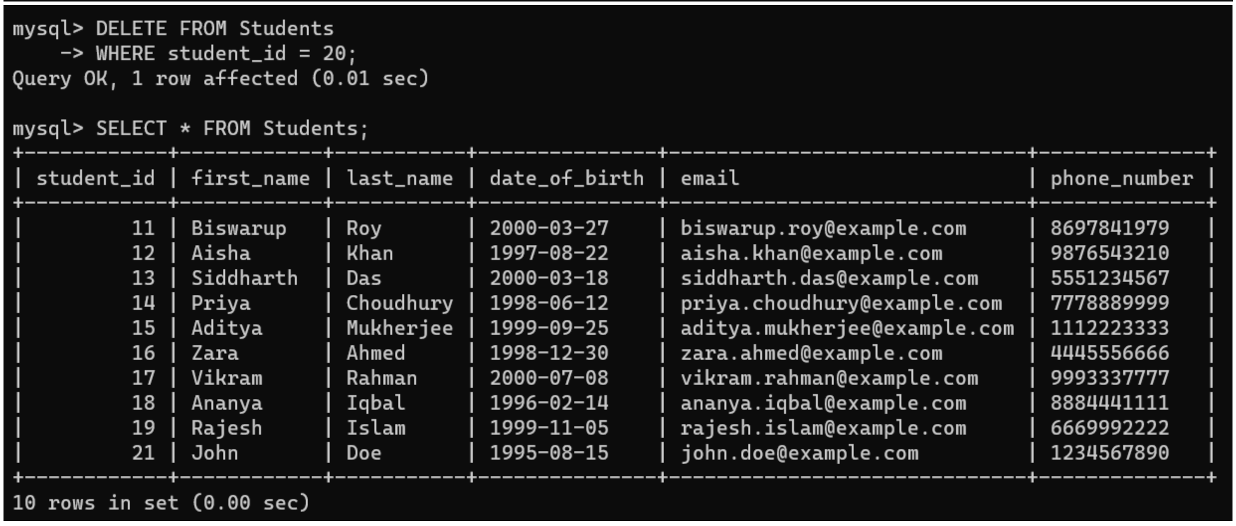
6. Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.



```
mysql> ALTER TABLE Payments
    -> ADD FOREIGN KEY (student_id)
    -> REFERENCES Students (student_id)
    -> ON DELETE CASCADE;
Query OK, 9 rows affected (0.05 sec)
Records: 9  Duplicates: 0  Warnings: 0
```

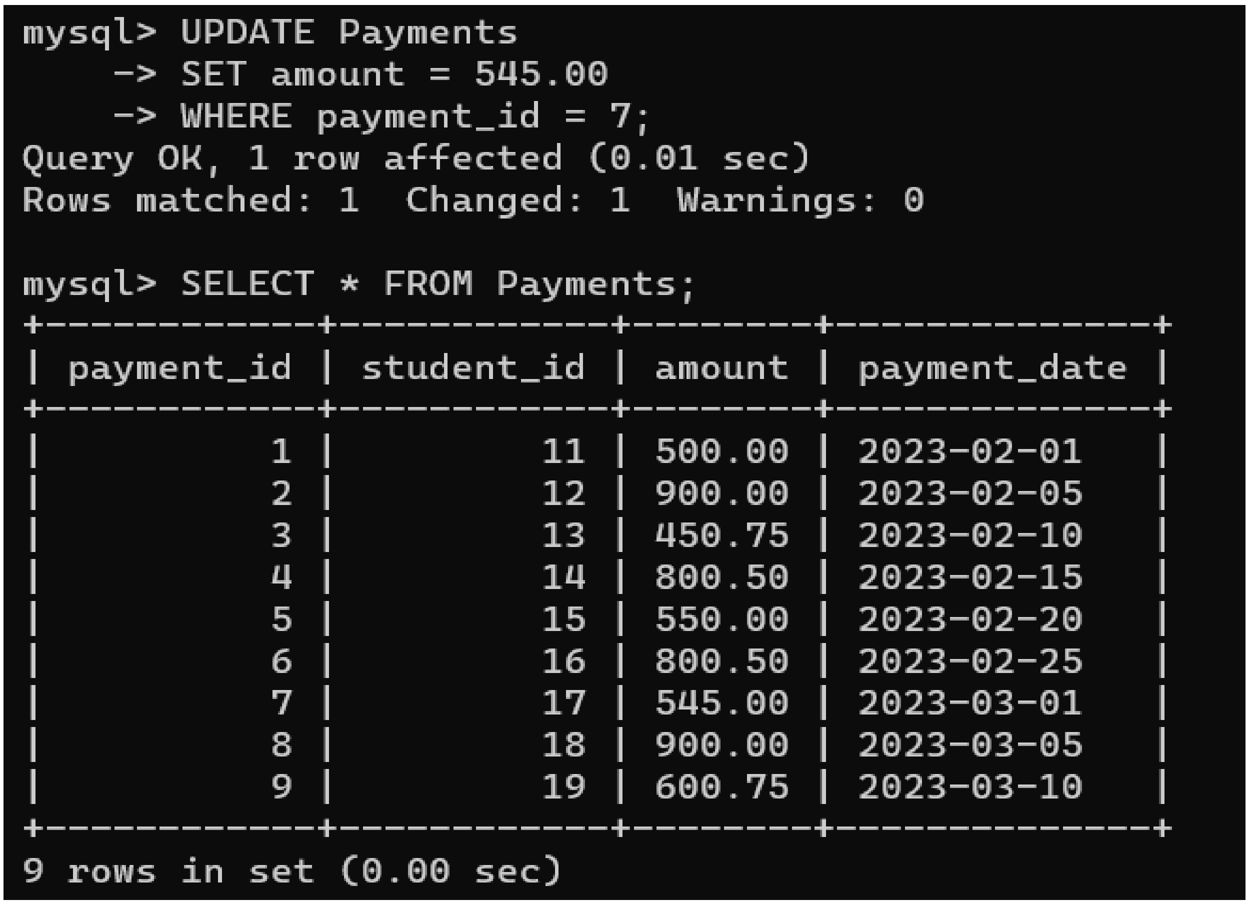


```
mysql> DELETE FROM Enrollments
    -> WHERE student_id = 20;
Query OK, 1 row affected (0.01 sec)
```



```
mysql> DELETE FROM Students
    -> WHERE student_id = 20;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Students;
+------------+------------+-----------+---------------+------------------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                        | phone_number |
+------------+------------+-----------+---------------+------------------------------+--------------+
|         11 | Biswarup   | Roy       | 2000-03-27    | biswarup.roy@example.com     | 8697841979   |
|         12 | Aisha      | Khan      | 1997-08-22    | aisha.khan@example.com       | 9876543210   |
|         13 | Siddharth  | Das       | 2000-03-18    | siddharth.das@example.com    | 5551234567   |
|         14 | Priya      | Choudhury | 1998-06-12    | priya.choudhury@example.com  | 7778889999   |
|         15 | Aditya     | Mukherjee | 1999-09-25    | aditya.mukherjee@example.com | 1112223333   |
|         16 | Zara       | Ahmed     | 1998-12-30    | zara.ahmed@example.com       | 4445556666   |
|         17 | Vikram     | Rahman    | 2000-07-08    | vikram.rahman@example.com    | 9993337777   |
|         18 | Ananya     | Iqbal     | 1996-02-14    | ananya.iqbal@example.com     | 8884441111   |
|         19 | Rajesh     | Islam     | 1999-11-05    | rajesh.islam@example.com     | 6669992222   |
|         21 | John       | Doe       | 1995-08-15    | john.doe@example.com         | 1234567890   |
+------------+------------+-----------+---------------+------------------------------+--------------+
10 rows in set (0.00 sec)
```

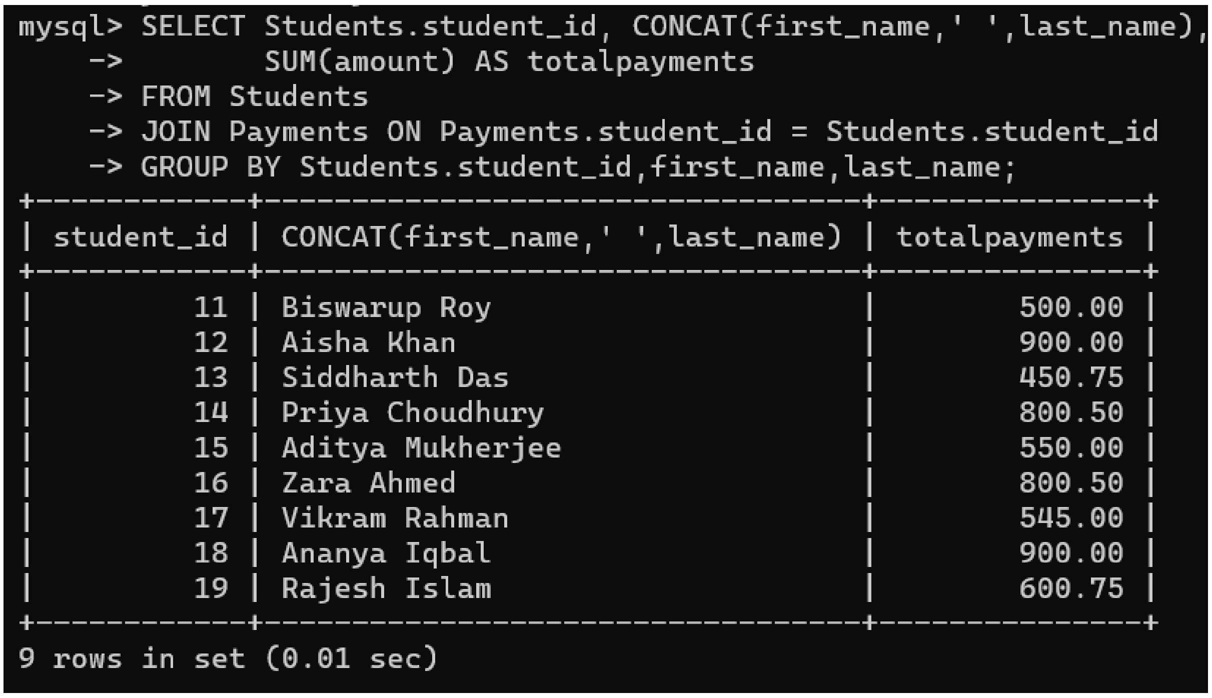
7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.



```
mysql> UPDATE Payments
    -> SET amount = 545.00
    -> WHERE payment_id = 7;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|          1 |         11 | 500.00 | 2023-02-01   |
|          2 |         12 | 900.00 | 2023-02-05   |
|          3 |         13 | 450.75 | 2023-02-10   |
|          4 |         14 | 800.50 | 2023-02-15   |
|          5 |         15 | 550.00 | 2023-02-20   |
|          6 |         16 | 800.50 | 2023-02-25   |
|          7 |         17 | 545.00 | 2023-03-01   |
|          8 |         18 | 900.00 | 2023-03-05   |
|          9 |         19 | 600.75 | 2023-03-10   |
+------------+------------+--------+--------------+
9 rows in set (0.00 sec)
```

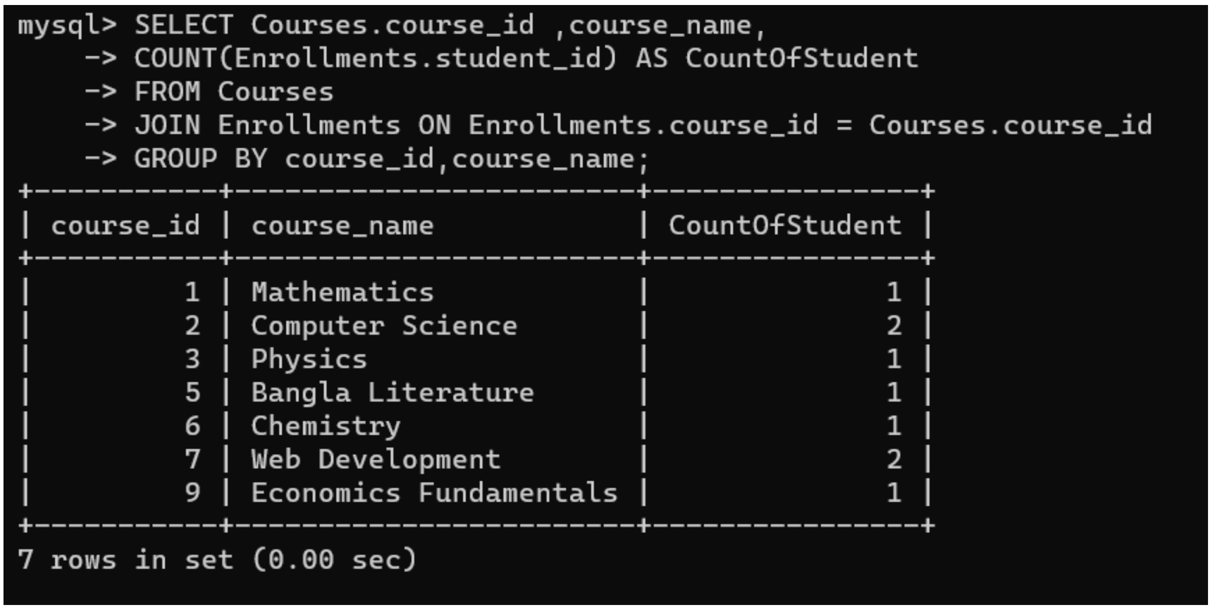
Task 3: Aggregate functions, Having, Order By, GroupBy and Joins

1. Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.



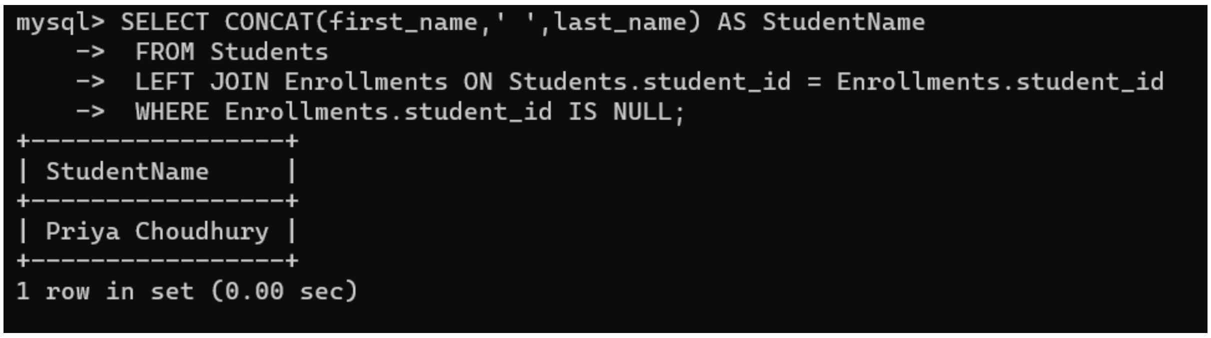
```
mysql> SELECT Students.student_id, CONCAT(first_name,' ',last_name),
    ->        SUM(amount) AS totalpayments
    -> FROM Students
    -> JOIN Payments ON Payments.student_id = Students.student_id
    -> GROUP BY Students.student_id,first_name,last_name;
+------------+----------------------------------+---------------+
| student_id | CONCAT(first_name,' ',last_name) | totalpayments |
+------------+----------------------------------+---------------+
|         11 | Biswarup Roy                     |        500.00 |
|         12 | Aisha Khan                       |        900.00 |
|         13 | Siddharth Das                    |        450.75 |
|         14 | Priya Choudhury                  |        800.50 |
|         15 | Aditya Mukherjee                 |        550.00 |
|         16 | Zara Ahmed                       |        800.50 |
|         17 | Vikram Rahman                    |        545.00 |
|         18 | Ananya Iqbal                     |        900.00 |
|         19 | Rajesh Islam                     |        600.75 |
+------------+----------------------------------+---------------+
9 rows in set (0.01 sec)
```

2. Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.



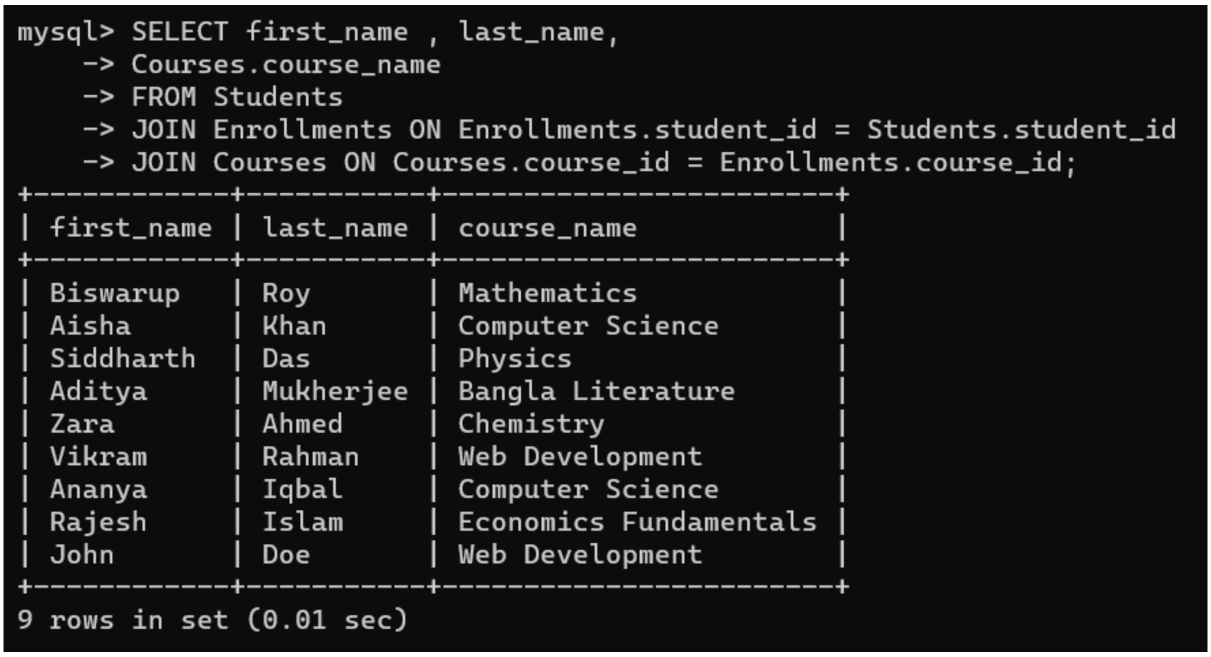
```
mysql> SELECT Courses.course_id ,course_name,
    -> COUNT(Enrollments.student_id) AS CountOfStudent
    -> FROM Courses
    -> JOIN Enrollments ON Enrollments.course_id = Courses.course_id
    -> GROUP BY course_id,course_name;
+-----------+------------------------+----------------+
| course_id | course_name            | CountOfStudent |
+-----------+------------------------+----------------+
|         1 | Mathematics            |              1 |
|         2 | Computer Science       |              2 |
|         3 | Physics                |              1 |
|         5 | Bangla Literature      |              1 |
|         6 | Chemistry              |              1 |
|         7 | Web Development        |              2 |
|         9 | Economics Fundamentals |              1 |
+-----------+------------------------+----------------+
7 rows in set (0.00 sec)
```

3. Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments.



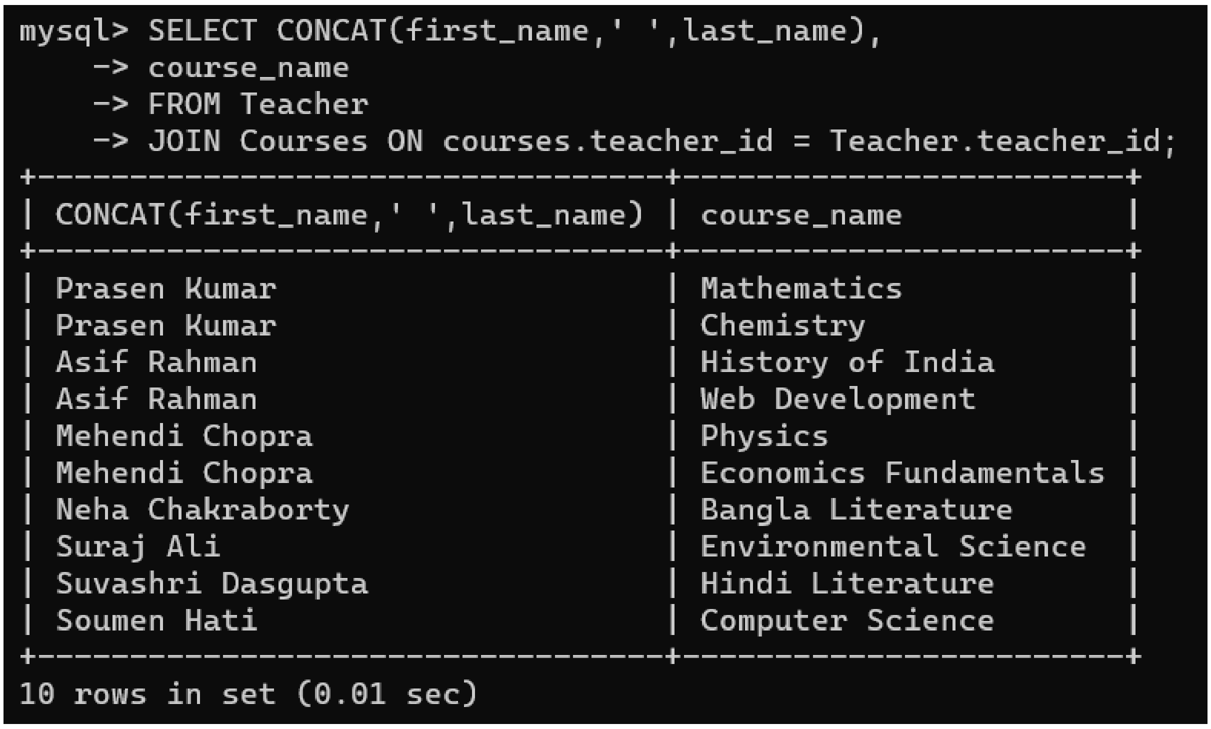
```
mysql> SELECT CONCAT(first_name,' ',last_name) AS StudentName
    ->  FROM Students
    ->  LEFT JOIN Enrollments ON Students.student_id = Enrollments.student_id
    ->  WHERE Enrollments.student_id IS NULL;
+-----------------+
| StudentName     |
+-----------------+
| Priya Choudhury |
+-----------------+
1 row in set (0.00 sec)
```

4. Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.



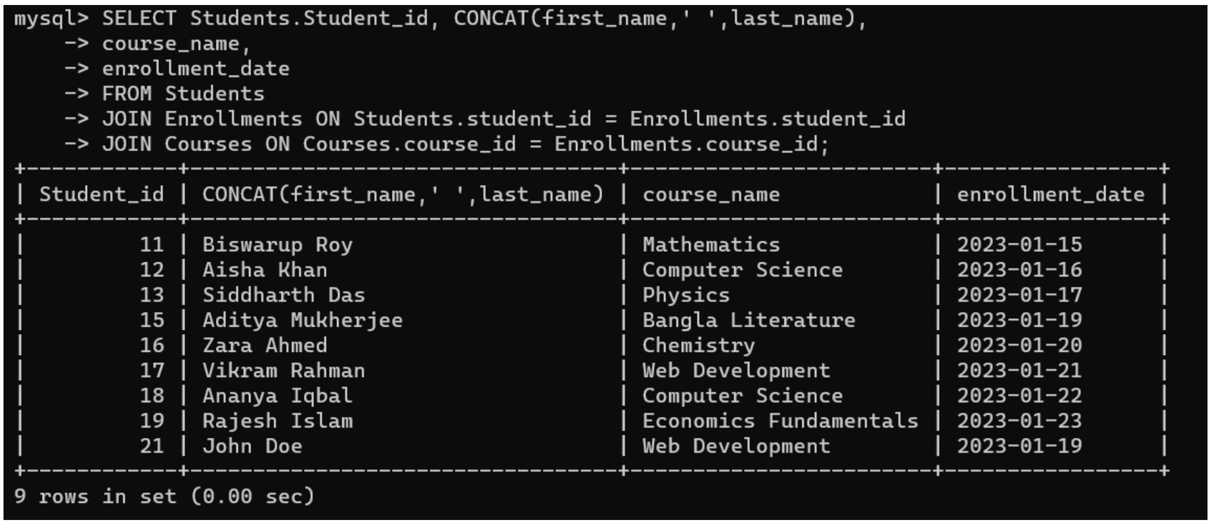
```
mysql> SELECT first_name , last_name,
    -> Courses.course_name
    -> FROM Students
    -> JOIN Enrollments ON Enrollments.student_id = Students.student_id
    -> JOIN Courses ON Courses.course_id = Enrollments.course_id;
+------------+-----------+------------------------+
| first_name | last_name | course_name            |
+------------+-----------+------------------------+
| Biswarup   | Roy       | Mathematics            |
| Aisha      | Khan      | Computer Science       |
| Siddharth  | Das       | Physics                |
| Aditya     | Mukherjee | Bangla Literature      |
| Zara       | Ahmed     | Chemistry              |
| Vikram     | Rahman    | Web Development        |
| Ananya     | Iqbal     | Computer Science       |
| Rajesh     | Islam     | Economics Fundamentals |
| John       | Doe       | Web Development        |
+------------+-----------+------------------------+
9 rows in set (0.01 sec)
```

5. Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.



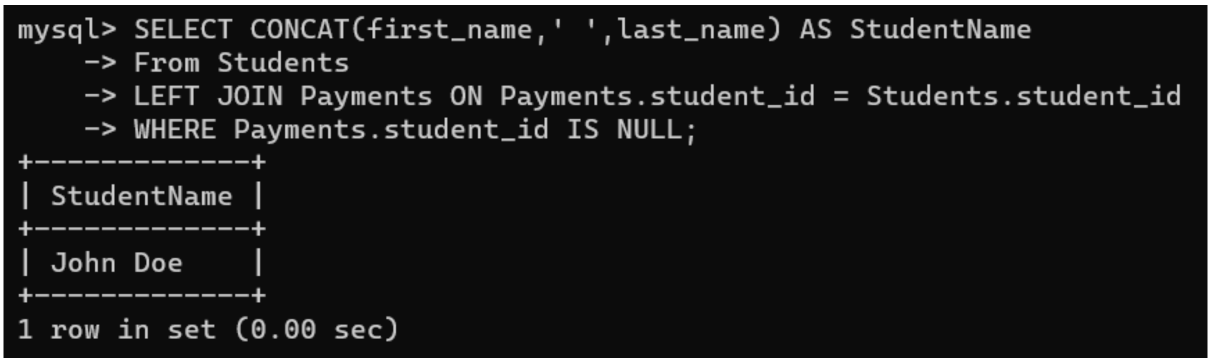
```
mysql> SELECT CONCAT(first_name,' ',last_name),
    -> course_name
    -> FROM Teacher
    -> JOIN Courses ON courses.teacher_id = Teacher.teacher_id;
+----------------------------------+------------------------+
| CONCAT(first_name,' ',last_name) | course_name            |
+----------------------------------+------------------------+
| Prasen Kumar                     | Mathematics            |
| Prasen Kumar                     | Chemistry              |
| Asif Rahman                      | History of India       |
| Asif Rahman                      | Web Development        |
| Mehendi Chopra                   | Physics                |
| Mehendi Chopra                   | Economics Fundamentals |
| Neha Chakraborty                 | Bangla Literature      |
| Suraj Ali                        | Environmental Science  |
| Suvashri Dasgupta                | Hindi Literature       |
| Soumen Hati                      | Computer Science       |
+----------------------------------+------------------------+
10 rows in set (0.01 sec)
```

6. Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.



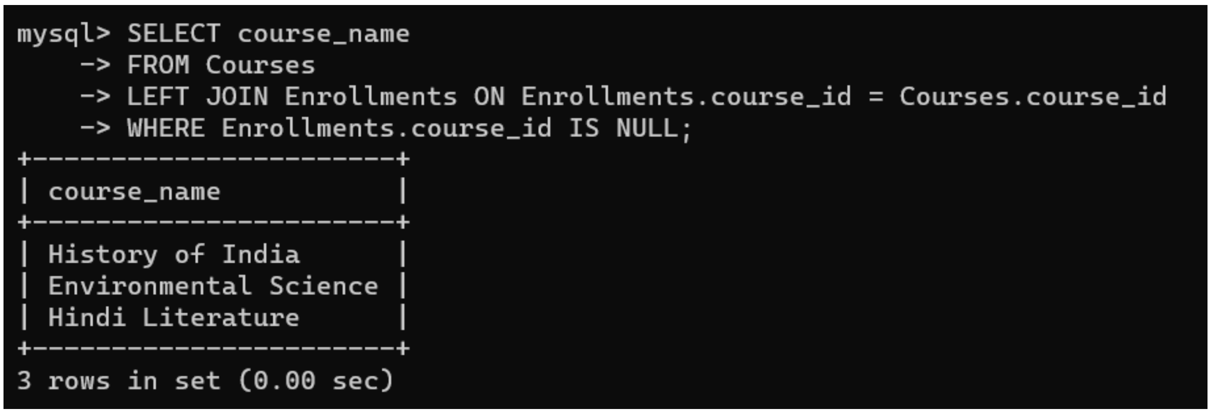
```
mysql> SELECT Students.Student_id, CONCAT(first_name,' ',last_name),
    -> course_name,
    -> enrollment_date
    -> FROM Students
    -> JOIN Enrollments ON Students.student_id = Enrollments.student_id
    -> JOIN Courses ON Courses.course_id = Enrollments.course_id;
+------------+----------------------------------+------------------------+-----------------+
| Student_id | CONCAT(first_name,' ',last_name) | course_name            | enrollment_date |
+------------+----------------------------------+------------------------+-----------------+
|         11 | Biswarup Roy                     | Mathematics            | 2023-01-15      |
|         12 | Aisha Khan                       | Computer Science       | 2023-01-16      |
|         13 | Siddharth Das                    | Physics                | 2023-01-17      |
|         15 | Aditya Mukherjee                 | Bangla Literature      | 2023-01-19      |
|         16 | Zara Ahmed                       | Chemistry              | 2023-01-20      |
|         17 | Vikram Rahman                    | Web Development        | 2023-01-21      |
|         18 | Ananya Iqbal                     | Computer Science       | 2023-01-22      |
|         19 | Rajesh Islam                     | Economics Fundamentals | 2023-01-23      |
|         21 | John Doe                         | Web Development        | 2023-01-19      |
+------------+----------------------------------+------------------------+-----------------+
9 rows in set (0.00 sec)
```

7. Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records.



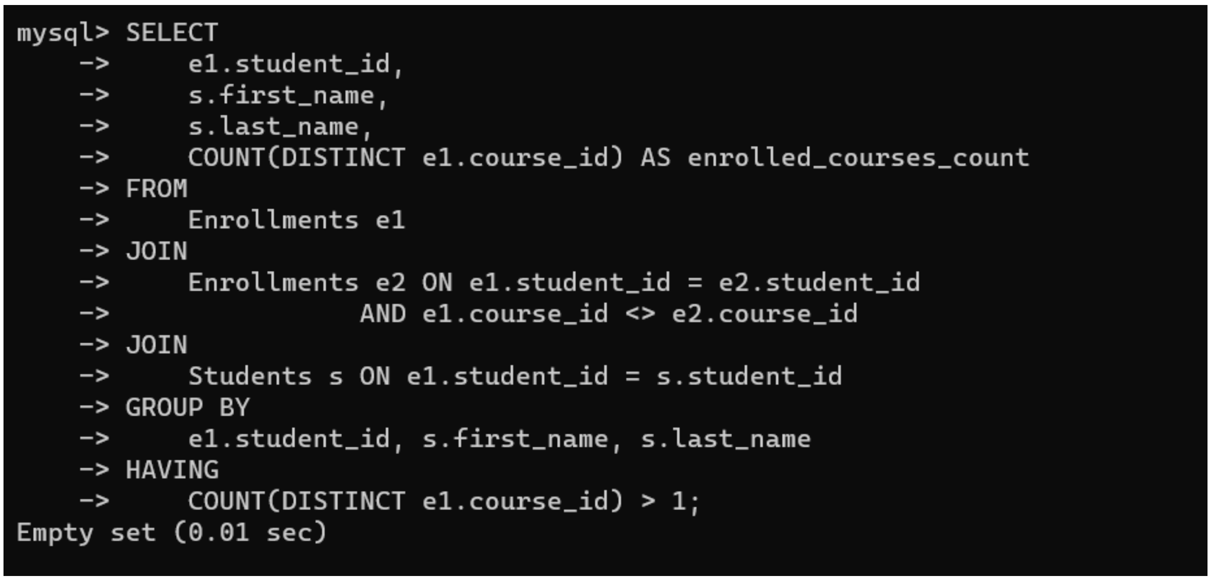
```
mysql> SELECT CONCAT(first_name,' ',last_name) AS StudentName
    -> From Students
    -> LEFT JOIN Payments ON Payments.student_id = Students.student_id
    -> WHERE Payments.student_id IS NULL;
+-------------+
| StudentName |
+-------------+
| John Doe    |
+-------------+
1 row in set (0.00 sec)
```

8. Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.



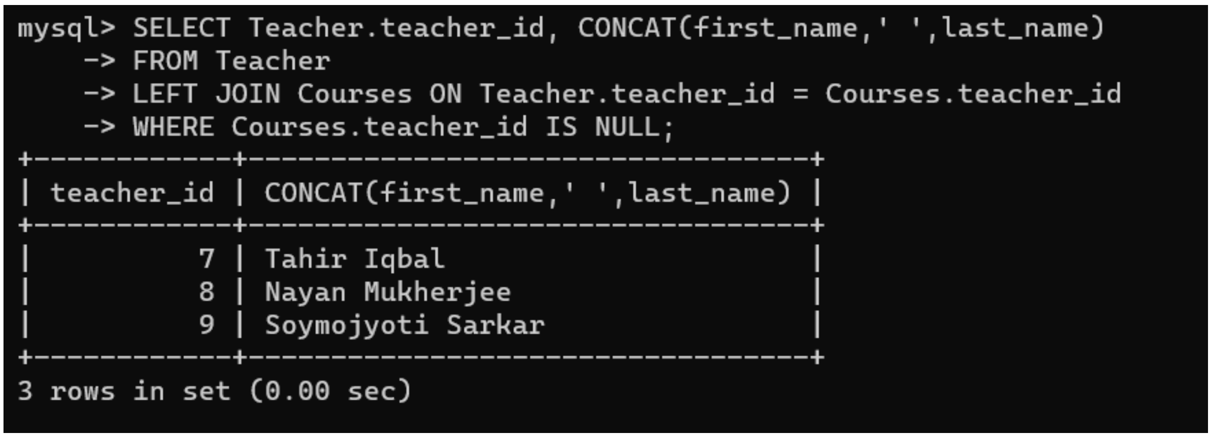
```
mysql> SELECT course_name
    -> FROM Courses
    -> LEFT JOIN Enrollments ON Enrollments.course_id = Courses.course_id
    -> WHERE Enrollments.course_id IS NULL;
+-----------------------+
| course_name           |
+-----------------------+
| History of India      |
| Environmental Science |
| Hindi Literature      |
+-----------------------+
3 rows in set (0.00 sec)
```

9. Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.



```
mysql> SELECT
    ->     e1.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     COUNT(DISTINCT e1.course_id) AS enrolled_courses_count
    -> FROM
    ->     Enrollments e1
    -> JOIN
    ->     Enrollments e2 ON e1.student_id = e2.student_id
    ->                 AND e1.course_id <> e2.course_id
    -> JOIN
    ->     Students s ON e1.student_id = s.student_id
    -> GROUP BY
    ->     e1.student_id, s.first_name, s.last_name
    -> HAVING
    ->     COUNT(DISTINCT e1.course_id) > 1;
Empty set (0.01 sec)
```

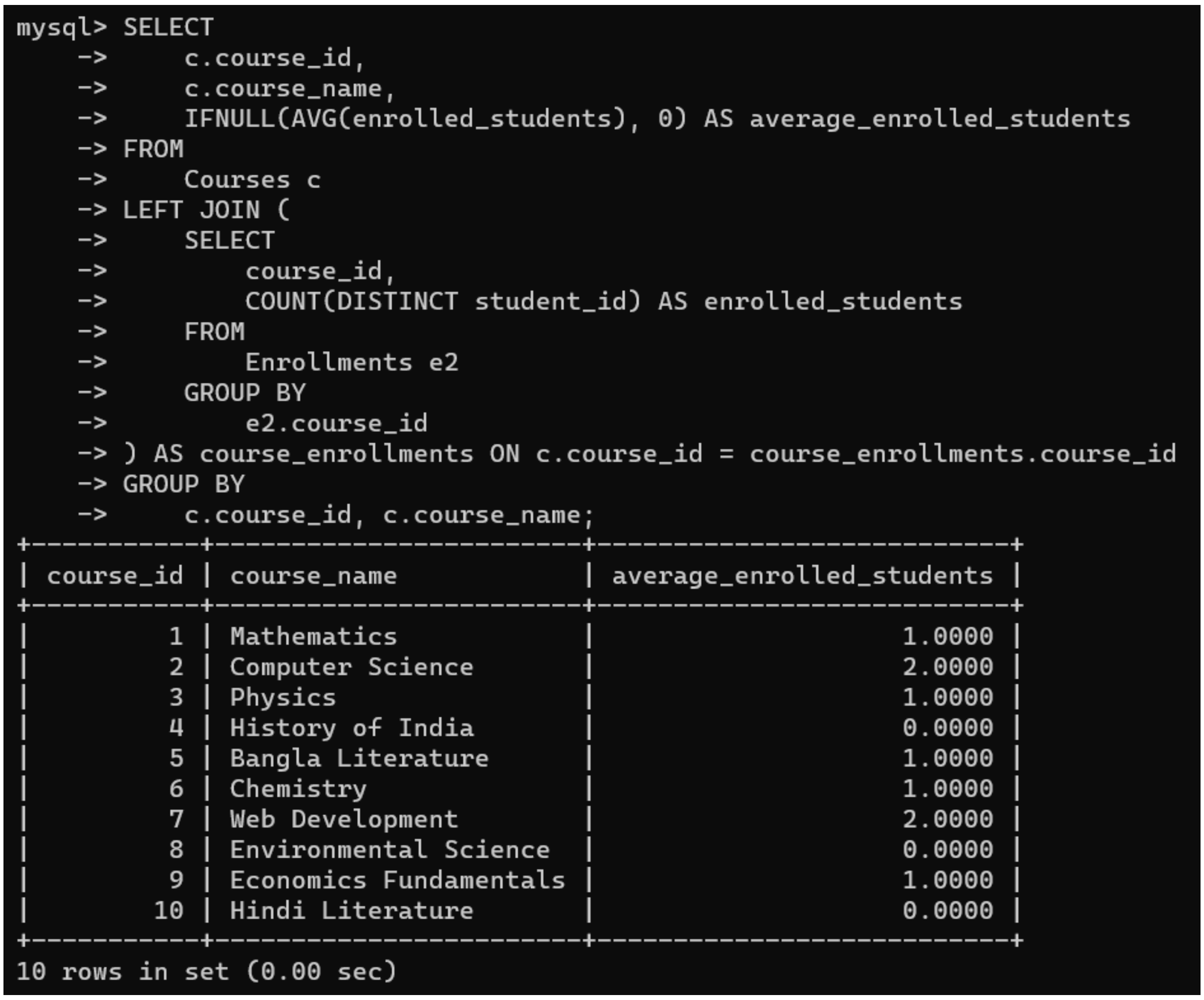
10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.



```
mysql> SELECT Teacher.teacher_id, CONCAT(first_name,' ',last_name)
    -> FROM Teacher
    -> LEFT JOIN Courses ON Teacher.teacher_id = Courses.teacher_id
    -> WHERE Courses.teacher_id IS NULL;
+------------+----------------------------------+
| teacher_id | CONCAT(first_name,' ',last_name) |
+------------+----------------------------------+
|          7 | Tahir Iqbal                      |
|          8 | Nayan Mukherjee                  |
|          9 | Soymojyoti Sarkar                |
+------------+----------------------------------+
3 rows in set (0.00 sec)
```

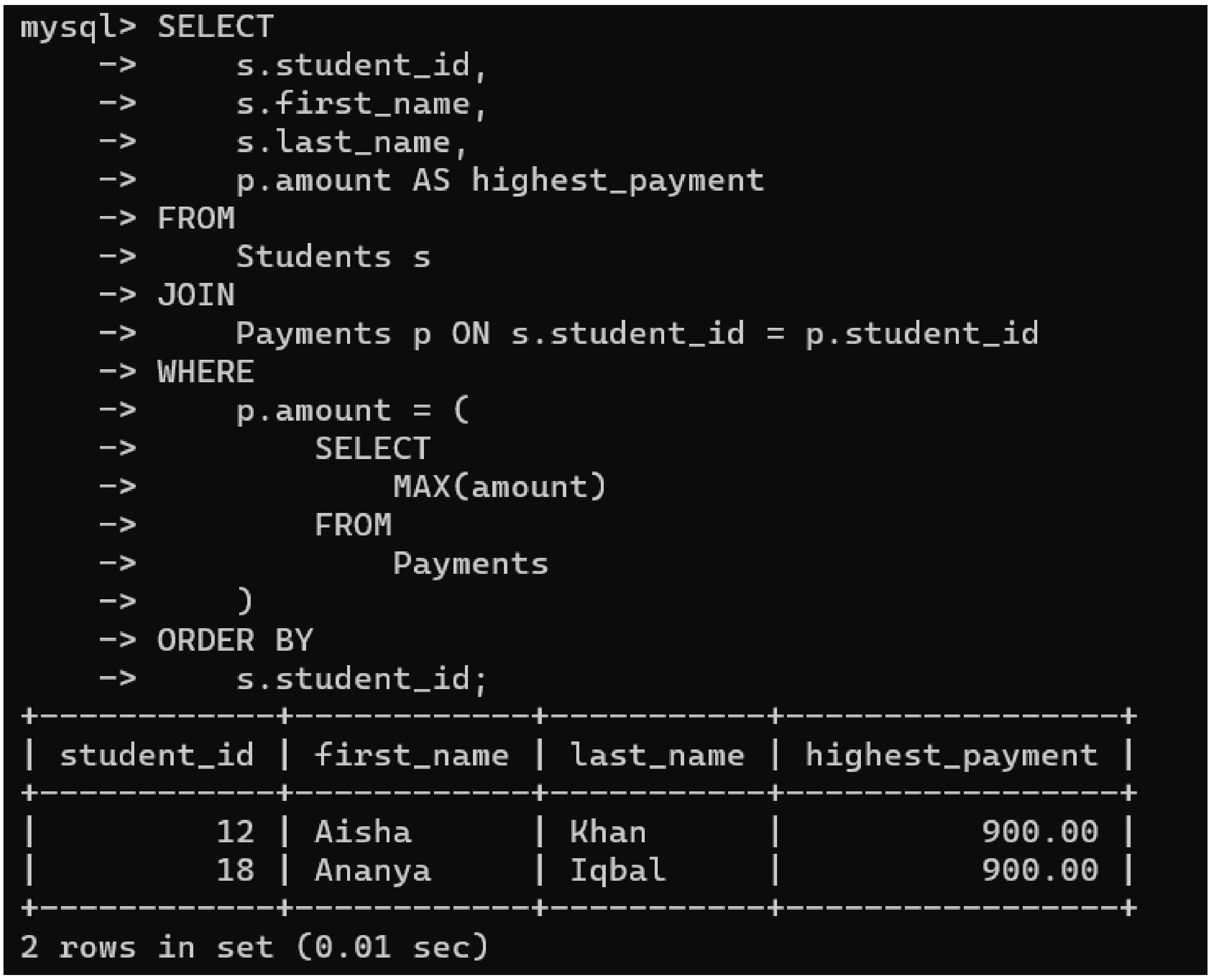
Task 4: Subquery and its type

1. Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.



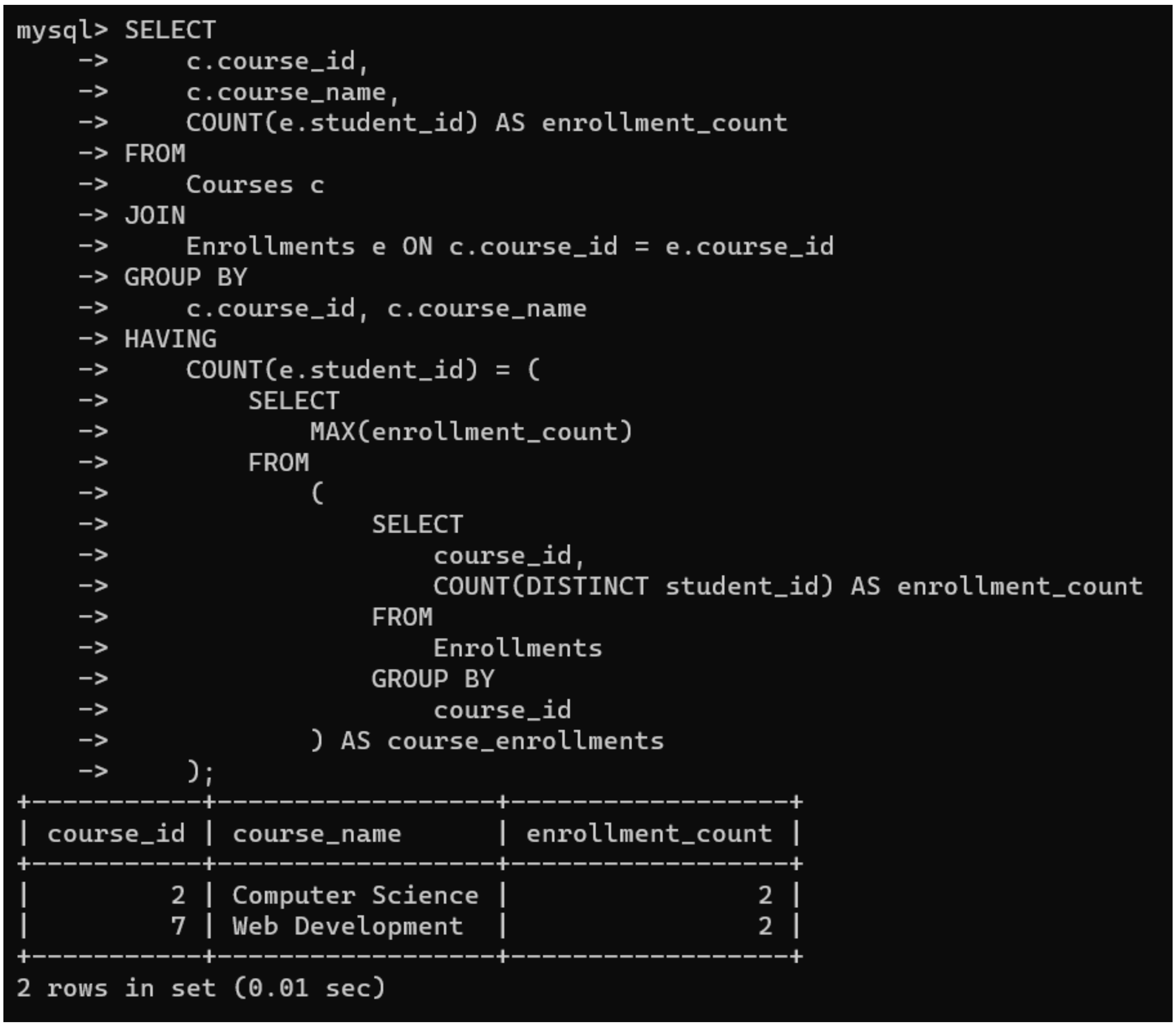
```
mysql> SELECT
    ->     c.course_id,
    ->     c.course_name,
    ->     IFNULL(AVG(enrolled_students), 0) AS average_enrolled_students
    -> FROM
    ->     Courses c
    -> LEFT JOIN (
    ->     SELECT
    ->         course_id,
    ->         COUNT(DISTINCT student_id) AS enrolled_students
    ->     FROM
    ->         Enrollments e2
    ->     GROUP BY
    ->         e2.course_id
    -> ) AS course_enrollments ON c.course_id = course_enrollments.course_id
    -> GROUP BY
    ->     c.course_id, c.course_name;
+-----------+------------------------+---------------------------+
| course_id | course_name            | average_enrolled_students |
+-----------+------------------------+---------------------------+
|         1 | Mathematics            |                    1.0000 |
|         2 | Computer Science       |                    2.0000 |
|         3 | Physics                |                    1.0000 |
|         4 | History of India       |                    0.0000 |
|         5 | Bangla Literature      |                    1.0000 |
|         6 | Chemistry              |                    1.0000 |
|         7 | Web Development        |                    2.0000 |
|         8 | Environmental Science  |                    0.0000 |
|         9 | Economics Fundamentals |                    1.0000 |
|        10 | Hindi Literature       |                    0.0000 |
+-----------+------------------------+---------------------------+
10 rows in set (0.00 sec)
```

2. Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.



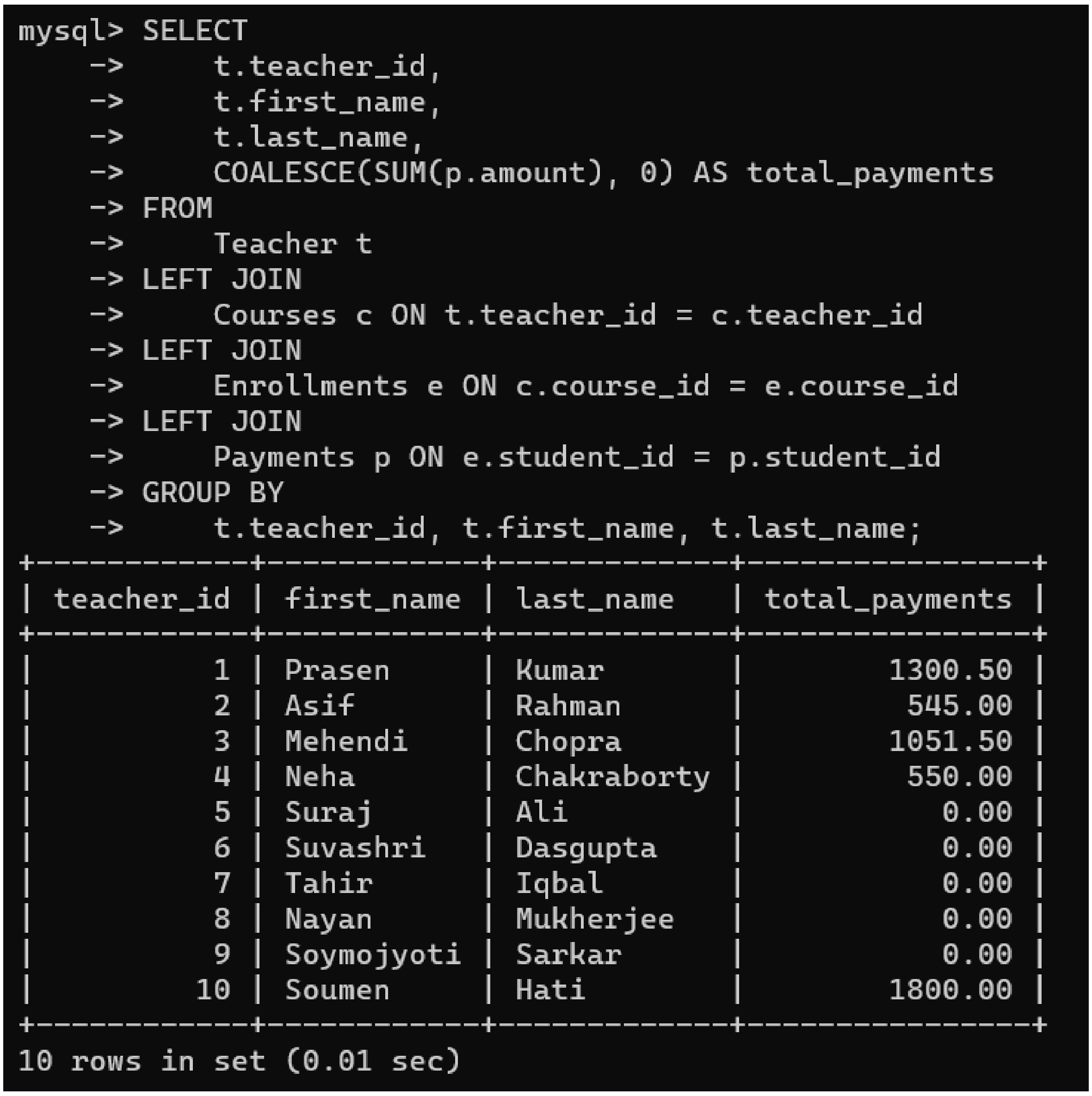
```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     p.amount AS highest_payment
    -> FROM
    ->     Students s
    -> JOIN
    ->     Payments p ON s.student_id = p.student_id
    -> WHERE
    ->     p.amount = (
    ->         SELECT
    ->             MAX(amount)
    ->         FROM
    ->             Payments
    ->     )
    -> ORDER BY
    ->     s.student_id;
+------------+------------+-----------+-----------------+
| student_id | first_name | last_name | highest_payment |
+------------+------------+-----------+-----------------+
|         12 | Aisha      | Khan      |          900.00 |
|         18 | Ananya     | Iqbal     |          900.00 |
+------------+------------+-----------+-----------------+
2 rows in set (0.01 sec)
```

3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.



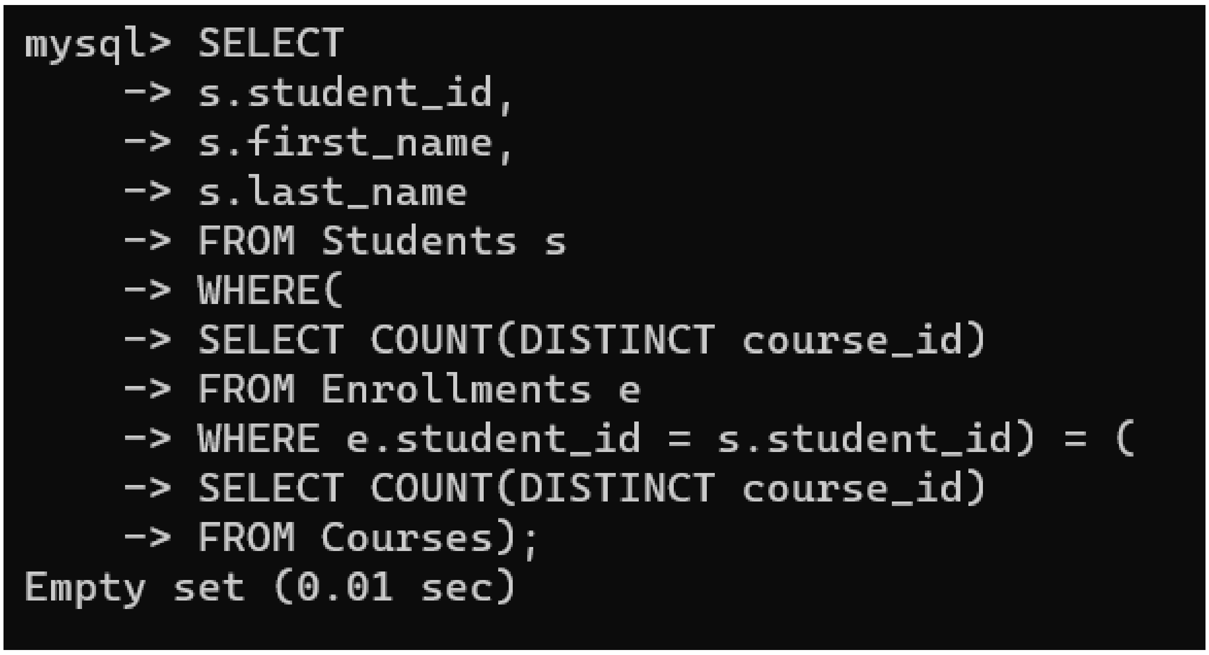
```
mysql> SELECT
    ->     c.course_id,
    ->     c.course_name,
    ->     COUNT(e.student_id) AS enrollment_count
    -> FROM
    ->     Courses c
    -> JOIN
    ->     Enrollments e ON c.course_id = e.course_id
    -> GROUP BY
    ->     c.course_id, c.course_name
    -> HAVING
    ->     COUNT(e.student_id) = (
    ->         SELECT
    ->             MAX(enrollment_count)
    ->         FROM
    ->             (
    ->                 SELECT
    ->                     course_id,
    ->                     COUNT(DISTINCT student_id) AS enrollment_count
    ->                 FROM
    ->                     Enrollments
    ->                 GROUP BY
    ->                     course_id
    ->             ) AS course_enrollments
    ->     );
+-----------+------------------+------------------+
| course_id | course_name      | enrollment_count |
+-----------+------------------+------------------+
|         2 | Computer Science |                2 |
|         7 | Web Development  |                2 |
+-----------+------------------+------------------+
2 rows in set (0.01 sec)
```

4.Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.



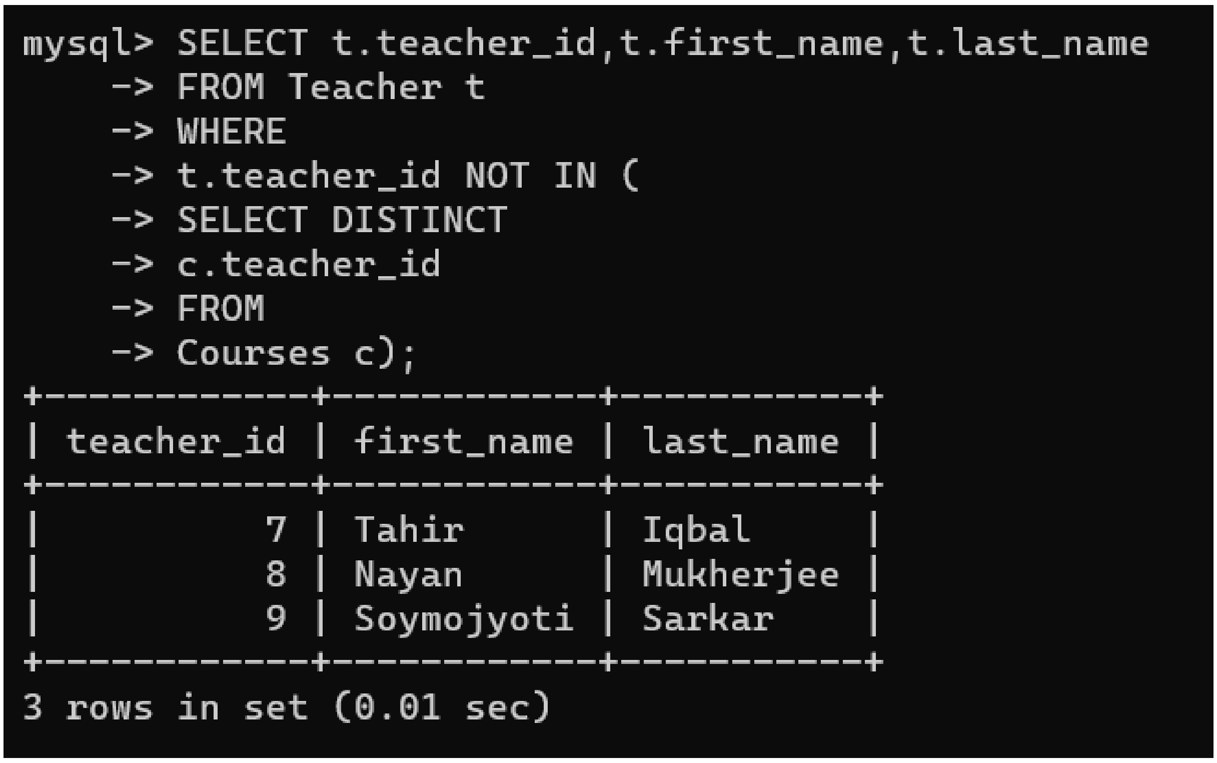
```
mysql> SELECT
    ->     t.teacher_id,
    ->     t.first_name,
    ->     t.last_name,
    ->     COALESCE(SUM(p.amount), 0) AS total_payments
    -> FROM
    ->     Teacher t
    -> LEFT JOIN
    ->     Courses c ON t.teacher_id = c.teacher_id
    -> LEFT JOIN
    ->     Enrollments e ON c.course_id = e.course_id
    -> LEFT JOIN
    ->     Payments p ON e.student_id = p.student_id
    -> GROUP BY
    ->     t.teacher_id, t.first_name, t.last_name;
+------------+------------+-------------+----------------+
| teacher_id | first_name | last_name   | total_payments |
+------------+------------+-------------+----------------+
|          1 | Prasen     | Kumar       |        1300.50 |
|          2 | Asif       | Rahman      |         545.00 |
|          3 | Mehendi    | Chopra      |        1051.50 |
|          4 | Neha       | Chakraborty |         550.00 |
|          5 | Suraj      | Ali         |           0.00 |
|          6 | Suvashri   | Dasgupta    |           0.00 |
|          7 | Tahir      | Iqbal       |           0.00 |
|          8 | Nayan      | Mukherjee   |           0.00 |
|          9 | Soymojyoti | Sarkar      |           0.00 |
|         10 | Soumen     | Hati        |        1800.00 |
+------------+------------+-------------+----------------+
10 rows in set (0.01 sec)
```

5. Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.



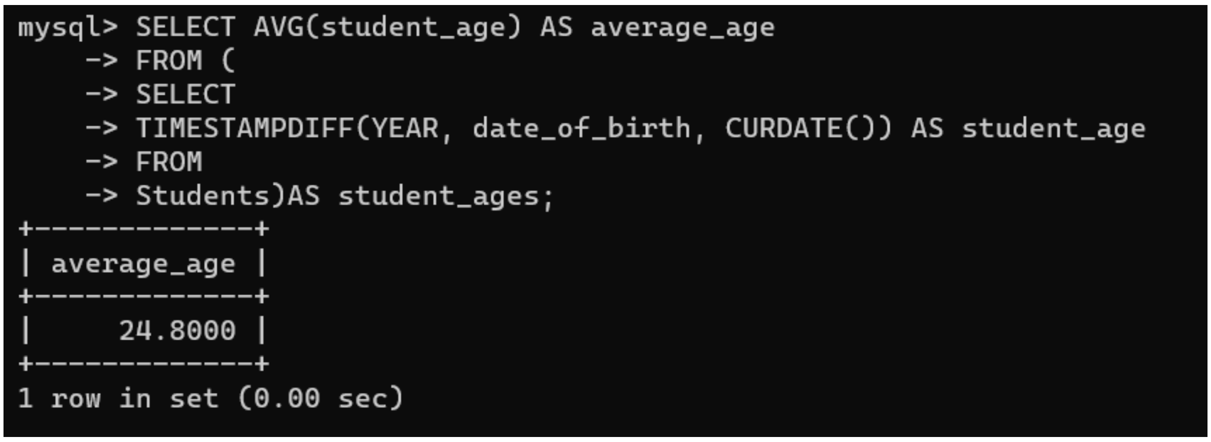
```
mysql> SELECT
    -> s.student_id,
    -> s.first_name,
    -> s.last_name
    -> FROM Students s
    -> WHERE(
    -> SELECT COUNT(DISTINCT course_id)
    -> FROM Enrollments e
    -> WHERE e.student_id = s.student_id) = (
    -> SELECT COUNT(DISTINCT course_id)
    -> FROM Courses);
Empty set (0.01 sec)
```

6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.



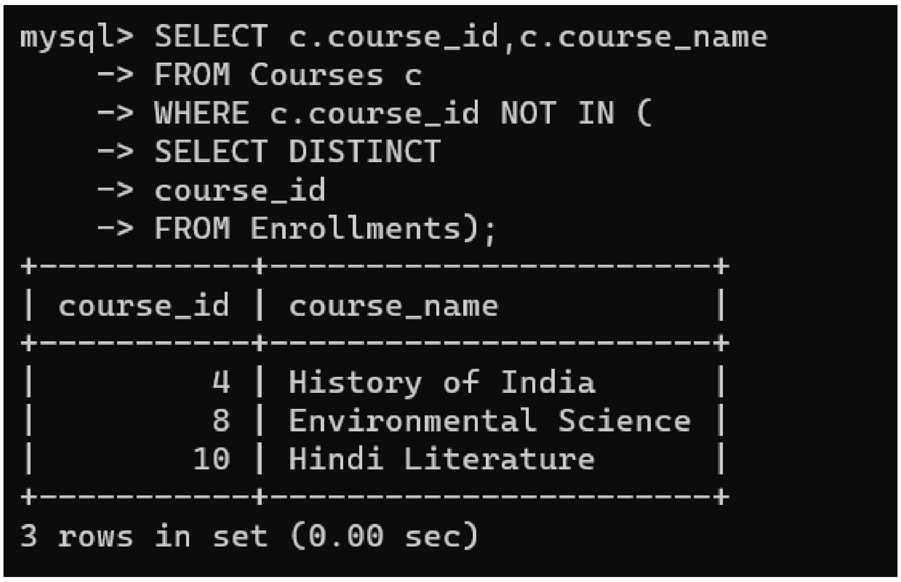
```
mysql> SELECT t.teacher_id,t.first_name,t.last_name
    -> FROM Teacher t
    -> WHERE
    -> t.teacher_id NOT IN (
    -> SELECT DISTINCT
    -> c.teacher_id
    -> FROM
    -> Courses c);
+------------+------------+-----------+
| teacher_id | first_name | last_name |
+------------+------------+-----------+
|          7 | Tahir      | Iqbal     |
|          8 | Nayan      | Mukherjee |
|          9 | Soymojyoti | Sarkar    |
+------------+------------+-----------+
3 rows in set (0.01 sec)
```

7. Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.



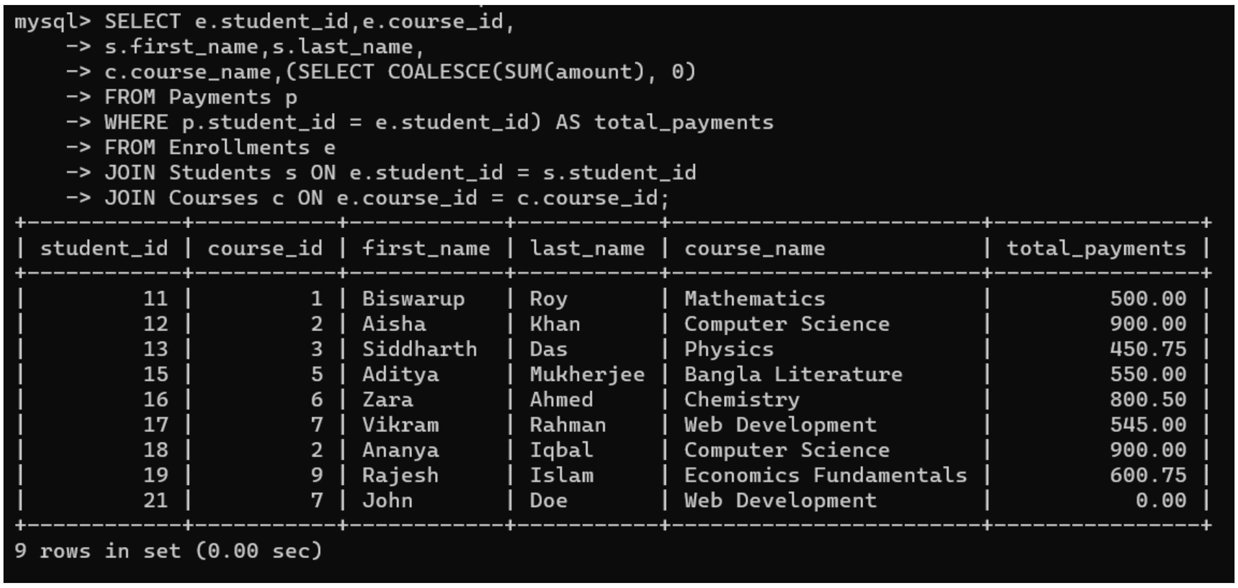
```
mysql> SELECT AVG(student_age) AS average_age
    -> FROM (
    -> SELECT
    -> TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS student_age
    -> FROM
    -> Students)AS student_ages;
+-------------+
| average_age |
+-------------+
|     24.8000 |
+-------------+
1 row in set (0.00 sec)
```

8. Identify courses with no enrollments. Use subqueries to find courses without enrollment records.



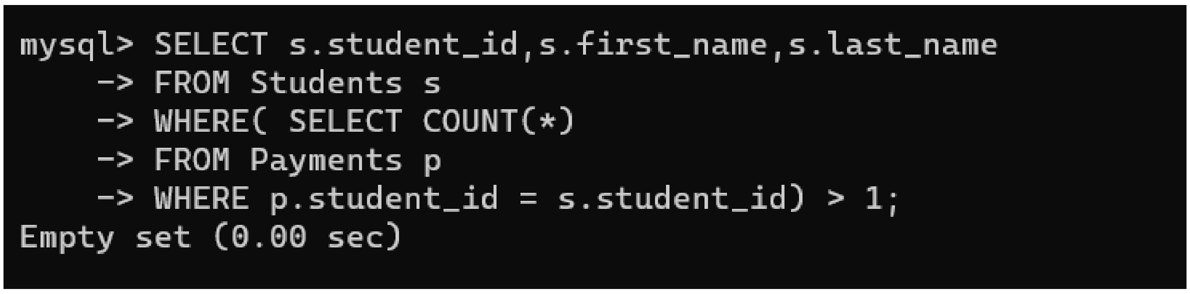
```
mysql> SELECT c.course_id,c.course_name
    -> FROM Courses c
    -> WHERE c.course_id NOT IN (
    -> SELECT DISTINCT
    -> course_id
    -> FROM Enrollments);
+-----------+-----------------------+
| course_id | course_name           |
+-----------+-----------------------+
|         4 | History of India      |
|         8 | Environmental Science |
|        10 | Hindi Literature      |
+-----------+-----------------------+
3 rows in set (0.00 sec)
```

9. Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.



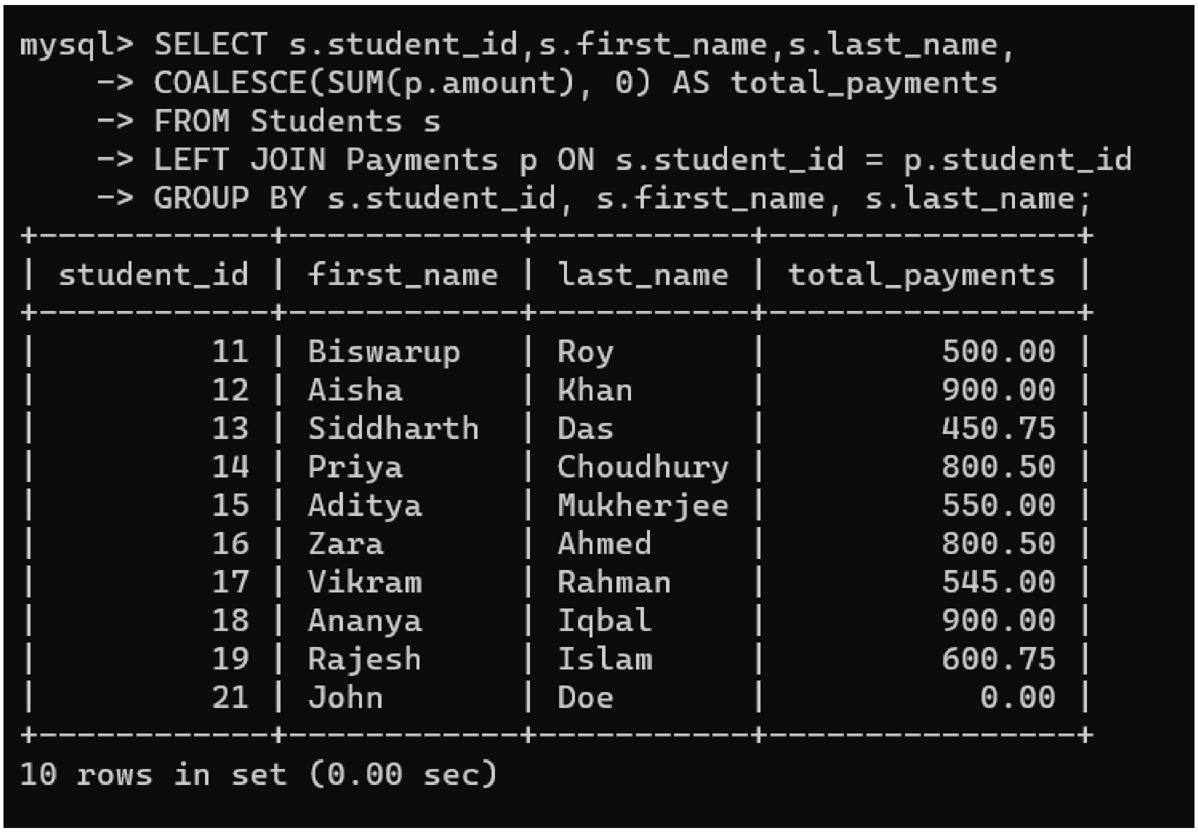
```
mysql> SELECT e.student_id,e.course_id,
    -> s.first_name,s.last_name,
    -> c.course_name,(SELECT COALESCE(SUM(amount), 0)
    -> FROM Payments p
    -> WHERE p.student_id = e.student_id) AS total_payments
    -> FROM Enrollments e
    -> JOIN Students s ON e.student_id = s.student_id
    -> JOIN Courses c ON e.course_id = c.course_id;
+------------+-----------+------------+-----------+------------------------+----------------+
| student_id | course_id | first_name | last_name | course_name            | total_payments |
+------------+-----------+------------+-----------+------------------------+----------------+
|         11 |         1 | Biswarup   | Roy       | Mathematics            |         500.00 |
|         12 |         2 | Aisha      | Khan      | Computer Science       |         900.00 |
|         13 |         3 | Siddharth  | Das       | Physics                |         450.75 |
|         15 |         5 | Aditya     | Mukherjee | Bangla Literature      |         550.00 |
|         16 |         6 | Zara       | Ahmed     | Chemistry              |         800.50 |
|         17 |         7 | Vikram     | Rahman    | Web Development        |         545.00 |
|         18 |         2 | Ananya     | Iqbal     | Computer Science       |         900.00 |
|         19 |         9 | Rajesh     | Islam     | Economics Fundamentals |         600.75 |
|         21 |         7 | John       | Doe       | Web Development        |           0.00 |
+------------+-----------+------------+-----------+------------------------+----------------+
9 rows in set (0.00 sec)
```

10. Identify students who have made more than one payment. Use subqueries and aggregate functions to count payments per student and filter for those with counts greater than one.



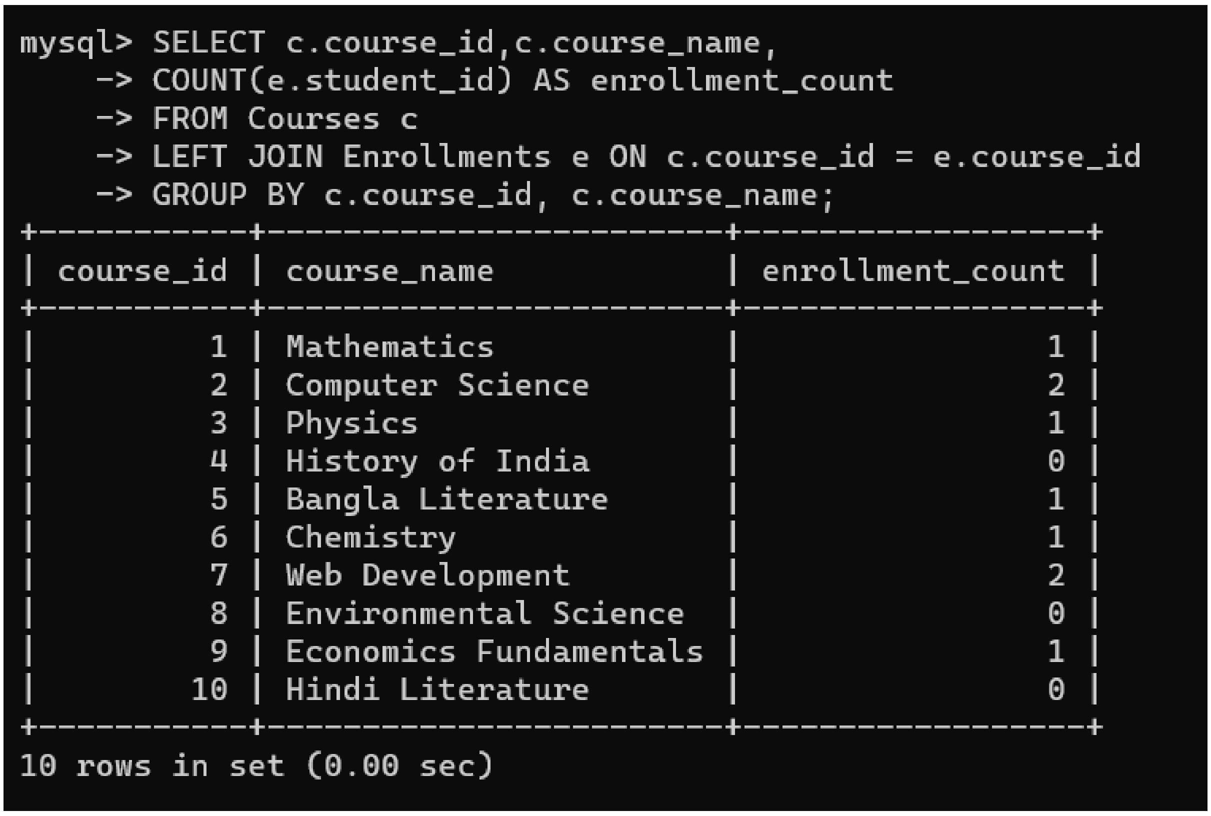
```
mysql> SELECT s.student_id,s.first_name,s.last_name
    -> FROM Students s
    -> WHERE( SELECT COUNT(*)
    -> FROM Payments p
    -> WHERE p.student_id = s.student_id) > 1;
Empty set (0.00 sec)
```

11. Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.



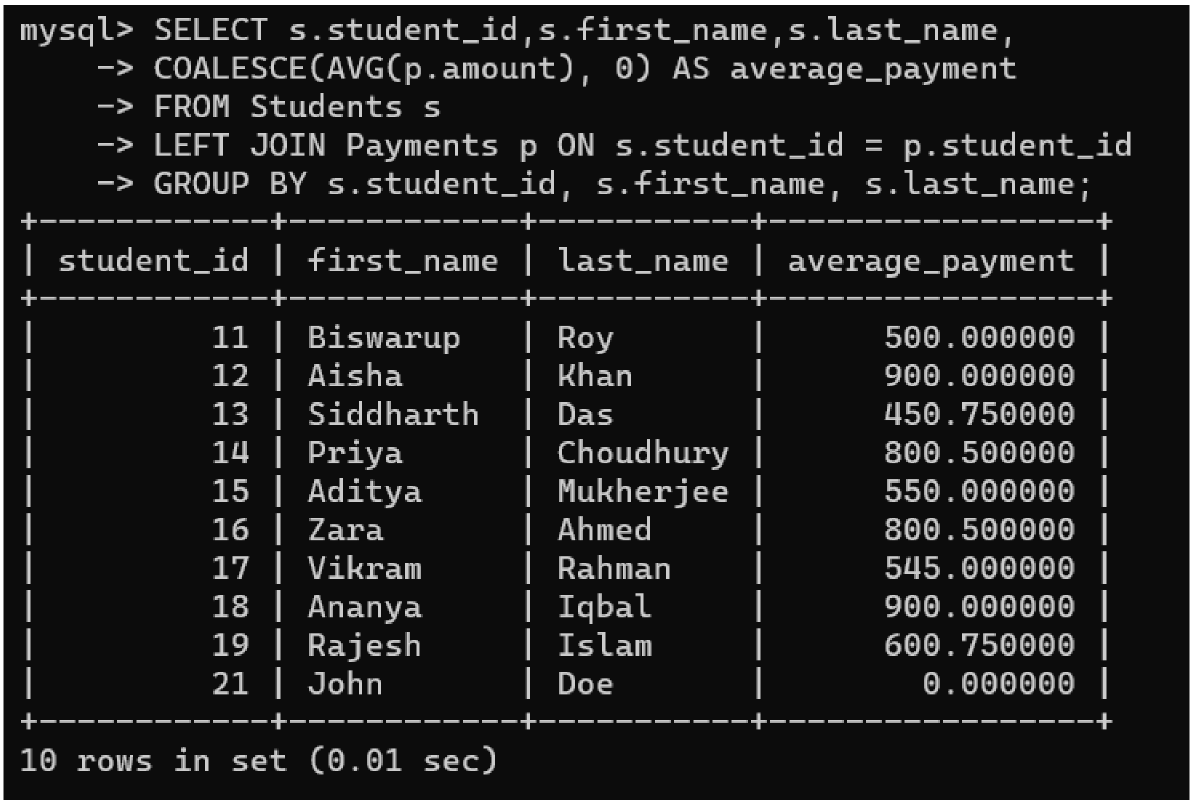
```
mysql> SELECT s.student_id,s.first_name,s.last_name,
    -> COALESCE(SUM(p.amount), 0) AS total_payments
    -> FROM Students s
    -> LEFT JOIN Payments p ON s.student_id = p.student_id
    -> GROUP BY s.student_id, s.first_name, s.last_name;
+------------+------------+-----------+----------------+
| student_id | first_name | last_name | total_payments |
+------------+------------+-----------+----------------+
|         11 | Biswarup   | Roy       |         500.00 |
|         12 | Aisha      | Khan      |         900.00 |
|         13 | Siddharth  | Das       |         450.75 |
|         14 | Priya      | Choudhury |         800.50 |
|         15 | Aditya     | Mukherjee |         550.00 |
|         16 | Zara       | Ahmed     |         800.50 |
|         17 | Vikram     | Rahman    |         545.00 |
|         18 | Ananya     | Iqbal     |         900.00 |
|         19 | Rajesh     | Islam     |         600.75 |
|         21 | John       | Doe       |           0.00 |
+------------+------------+-----------+----------------+
10 rows in set (0.00 sec)
```

12. Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.



```
mysql> SELECT c.course_id,c.course_name,
    -> COUNT(e.student_id) AS enrollment_count
    -> FROM Courses c
    -> LEFT JOIN Enrollments e ON c.course_id = e.course_id
    -> GROUP BY c.course_id, c.course_name;
+-----------+------------------------+------------------+
| course_id | course_name            | enrollment_count |
+-----------+------------------------+------------------+
|         1 | Mathematics            |                1 |
|         2 | Computer Science       |                2 |
|         3 | Physics                |                1 |
|         4 | History of India       |                0 |
|         5 | Bangla Literature      |                1 |
|         6 | Chemistry              |                1 |
|         7 | Web Development        |                2 |
|         8 | Environmental Science  |                0 |
|         9 | Economics Fundamentals |                1 |
|        10 | Hindi Literature       |                0 |
+-----------+------------------------+------------------+
10 rows in set (0.00 sec)
```

13. Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.



```
mysql> SELECT s.student_id,s.first_name,s.last_name,
    -> COALESCE(AVG(p.amount), 0) AS average_payment
    -> FROM Students s
    -> LEFT JOIN Payments p ON s.student_id = p.student_id
    -> GROUP BY s.student_id, s.first_name, s.last_name;
+------------+------------+-----------+-----------------+
| student_id | first_name | last_name | average_payment |
+------------+------------+-----------+-----------------+
|         11 | Biswarup   | Roy       |      500.000000 |
|         12 | Aisha      | Khan      |      900.000000 |
|         13 | Siddharth  | Das       |      450.750000 |
|         14 | Priya      | Choudhury |      800.500000 |
|         15 | Aditya     | Mukherjee |      550.000000 |
|         16 | Zara       | Ahmed     |      800.500000 |
|         17 | Vikram     | Rahman    |      545.000000 |
|         18 | Ananya     | Iqbal     |      900.000000 |
|         19 | Rajesh     | Islam     |      600.750000 |
|         21 | John       | Doe       |        0.000000 |
+------------+------------+-----------+-----------------+
10 rows in set (0.01 sec)
```